A Hybrid Neural Network-Fuzzy Logic Architecture for Multisensor Data Fusion in Target Tracking System

Xiu Wang[1] Jian Rong[1] Xiaochun Zhong [2]

[1]School of Physical Electronic, University of Electronic Science and Technology of China, Chengdu 610054, P. R. China

[2]Dept. of Applied Physics, Southwest Jiaotong University, Chengdu 610031, P. R. China

Abstract

In this work, a new multisensor data fusion architecture integrating neural network and fuzzy logic techniques is introduced, which has the ability of fast adjusting acceleration parameter and covariance of measurement noise of sensors. In this architecture, the neural network estimates acceleration and fuzzy logic adapts the covariance of measurement noise on-line and also offers degree of confidence of sensors for fusion. The results of simulation show that this new architecture can adjust maneuver parameter in nearly one sample time and change the covariance of measurement noise effectively.

Keywords: Neural network, Fuzzy logic, Multisensor data fusion, Target tracking

1. Introduction

Recent advances in sensor technology and distributed computational algorithms, together with demands for escalating operational requirements, have contributed to the growing desire to deploy multiple sensors in various surveillance and tactical missions. With the growing availability and the need for multisensor operations, however, there is a corresponding growth in the complexity of these systems, mainly arising from the diversity of the types of sensors used and the special problems that may be created such as asynchronous data arrivals and the need to order and correlate these data streams to exploit the inherent synergy available. Thus, the fundamental question of interest in these scenarios is how to develop an efficient data fusion architecture that facilitates integrated processing of the large volumes of data arriving at typically high rates to enable the needed decision making, which in turn facilitates fully utilizing the capabilities of these sensors and realizing all the possible benefits from their deployment.

In order to fuse the data from different sensors, a lot of fusion architectures and models have been brought forward. During all of those, two models have received most of popularity, the interacting multiple model (IMM) [1] introduced in 1984 and Input Estimation (IE) [2] model put forward by Y. T. CHAN, and ATC. HUN. But these two models still need to be improved. For example, when the target leads a maneuver of short term acceleration, IE model might loss the target, this problem was solved by Malur K. Sundareshan, who proposed to use the neural network to estimate the acceleration [3]. In 2003, Mark W. Owena1 and Allen R. Stubberud proposed NEKF IMM tracking [4] algorithm which addresses the disadvantage of IMM that using a high process noise model to hold a target through a maneuver with poor velocity and acceleration estimates.

When different models are discussed, online variety of measurement noise should also be taken into consideration. There are a lot of methods [5]-[7] trying to address this problem. One of those useful technologies is to use fuzzy logic [7] to adaptively adjust the covariance of measurement noise on-line.

Although the maneuver of short term acceleration and the variety of measurement noise have been discussed respectively, little work has been done to solve previous two issues in the same architecture. In that case, this study introduces a new distributed architecture integrating neural network and fuzzy logic. In this fusion system, the neural network takes data of different sensors as input, then exports estimate of acceleration to the Kalman filter and, at the same time, the fuzzy logic system which adjusts the covariance of measurement noise of sensors while providing the degree of confidence to fuse the results given by Kalman filters.

The remainder of this paper is organized as follows. Section II describes the details of this fusion architecture. In order to test the effectiveness of this architecture, in section III an illustrative example is

outlined and the results are discussed. Finally, a conclusion of this work is given in section IV.

2. Description of the new data fusion architecture

2.1. Kalman filter

The Kalman filter is an optimal recursive data processing algorithm [8] that provides a linear, unbiased, and minimum error variance estimate of the unknown state vector $x_k \in \Re^n$ at each instant k=1,2,…,(indexed by the subscripts) of a discrete-time controlled process described by linear stochastic difference equation:

$$x_{k+1} = A_k x_k + B_k u_k + w_k \quad (1)$$

$$z_k = H_k x_k + v_k \quad (2)$$

where x_k is an $n \times 1$ system state vector, A_k is an $n \times n$ transition matrix, u_k is an $l \times 1$ vector of the input forcing function, B_k is an $n \times l$ matrix, w_k is an $n \times 1$ process noise vector, z_k is an $m \times 1$ measurement vector, H_k is an $m \times n$ measurement matrix, and v_k is an $m \times 1$ measurement noise vector.

Both w_k and v_k are assumed to be uncorrelated zero-mean Gaussian white noise sequence with covariances,

$$E\{w_k w_i^T\} = \begin{cases} Q_k, & i = k \\ 0 & i \neq k \end{cases} \quad (3)$$

$$E\{v_k v_i^T\} = \begin{cases} R_k, & i = k \\ 0 & i \neq k \end{cases} \quad (4)$$

$$E\{w_k v_i^T\} = 0 \quad \text{for all } k \text{ and } i \quad (5)$$

where $E\{\cdot\}$ is the statistical expectation, superscript T denotes transpose, Q_k is the process noise covariance matrix and R_k is the measurement noise covariance matrix.

The Kalman filter algorithm can be organized in two groups of equations,

- Time update (or prediction) equations:

$$\hat{x}_{k+1}^- = A_k \hat{x}_k + B_k u_k \quad (6)$$

$$P_{k+1}^- = A_k P_k A_k^T + Q_k \quad (7)$$

These equations project, from time step k to step k+1, the current state and error covariance estimate to obtain the a priori (indicated by the super minus) estimates for the next time step.

- Measurement update (correction) equations:

$$K_k = P_k^- H_k^T [H_k P_k^- H_k^T + R_k]^{-1} \quad (8)$$

$$\hat{x}_k = \hat{x}_k^- + K[z_k - H_k \hat{x}_k^-] \quad (9)$$

$$P_k = [I - K_k H_k] P_k^- \quad (10)$$

These equations incorporate a new measurement into a priori estimate to obtain an improved a posteriori estimate.

In the above equations, $\hat{x}_k$ is an estimate of the system state vector x_k, and P_k is the covariance matrix corresponding to the state estimation error defined by,

$$P_k = E\{(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T\} \quad (11)$$

The term $H_k \hat{x}_k^-$ is the one-stage predicted output z_k, and $(z_k - H_k \hat{x}_k^-)$ is the one-stage prediction error sequence, also referred to as the innovation sequence or residual, generally denoted as r and defined as:

$$r_k = (z_k - H_k \hat{x}_k^-) \quad (12)$$

The innovation represents the additional information available to the filter as a consequence of the new observation z_k. The weighted innovation, $K_k[z_k - H_k \hat{x}_k^-]$, acts as a correction to the prediction estimate $\hat{x}_k^-$ to form the estimate $\hat{x}_k$; the weighting matrix K_k is commonly referred to as the filter gain or Kalman gain matrix.

The Kalman filter algorithm starts with initial conditions at k=0 being: $\hat{x}_0^-$, and P_0^-. With the progression of time, as new measurements z_k become available, the cycle estimation-correction of states and the corresponding error covariance can follow recursively ad infinitum.

2.2. Neural network

Artificial neural networks are emerging as very attractive alternatives to traditional methods (maximum likelihood techniques, nearest-neighbor classification, etc.) in the development of computer-based pattern classification algorithms since they can learn to perform the required classification without the assumption of probabilistic models for the input patterns. This area has witnessed an explosion of research in the recent past and one of the important results is based on the celebrated Kolmogorov theorem. This result states that any continuous nonlinear mapping can be approximated as closely as desired by a multilayered neural network with a feedforward topology and sigmoidal nonlinear functions [9]-[10].

The basic processing element (neuron) in these function approximating networks has an input-output characteristic that is obtained by forming a weighted sum of the several inputs received and producing an output that is a nonlinear function of this weighted sum, according to the relation:

$$y(t) = g\left[\sum_{i=1}^{m} w_i u_i(t)\right] \quad (13)$$

where $u_i(\cdot): \Re \to \Re$, $i = 1, 2, ..., m$, are the inputs; $y(\cdot): \Re \to \Re$ is output; and $w_i \in \Re$, $i = 1, 2, ..., m$, are the weights. Here $g(\cdot): \Re \to \Re$ is an appropriately selected nonlinear activation function that satisfies the following conditions:

- $xg(x) > 0$ for all $x \in \Re$ (first and third quadrant function).
- $\lim_{|x|\to\infty} g(x) = k\,\mathrm{sgn}(k), k > 0$ (saturation function).
- $g(x_1)/x_1 \geq g(x_2)/x_2$ for all $|x_1| \leq |x_2|$

Commonly used activation functions are the sigmoid characteristics $[e.g., g(x) = \tanh(\lambda x), or \quad g(x) = (1+e^{-x})^{-1}]$. The processing architecture of an illustrative multilayer feedforward network with one input layer, one output layer, and several hidden layers is shown in Fig. 1. In this architecture, the input layer has four nodes, which merely serve to fan out the incoming inputs to the nodes in the succeeding layer (viz., the first hidden layer) and the output layer has two nodes which merely combine the outputs from the nodes in the previous layer (viz., the last hidden layer). The hidden layers have arbitrary numbers of nodes that perform the nonlinear processing according to the rule just stated.

Of fundamental importance for the satisfactory training of the neural network is the selection of an appropriate set of input features. By using a sufficient number of features in the training process, it is possible to build very desirable fault tolerance properties (robustness) to the neural network processor. Furthermore, since the network architecture can be appropriately designed to accept as inputs data collected from a number of different sensors, data fusion can be naturally accomplished. Some input preprocessing may, however, be needed to modify the available data into a form that could be advantageously utilized as network inputs.

The ability of neural network to offer acceleration parameter on-line is proved in reference 3. According to different sensors, neural network accepts different inputs. Selection of appropriate features can be guided by observation that generally has three basic entities that help acquire a good estimate of target maneuver. These are (1) intensity of acceleration, (2) direction of tangential velocity, and (3) initial velocity at the time of acceleration. Take lidar for example, suppose the sensor of lidar can offer target position with two dimensions. Two inputs [3] are available to the neural network extracted from the raw sensor data:

- v_1 : intensity of acceleration, defined as follows:

$$v_1(k) = \frac{r_x^2(k)}{S_{xx}(k)} + \frac{r_y^2(k)}{S_{yy}(k)} \quad (14)$$

where $r(k) = [r_x r_y]$ is defined in Eq. 12, and $S_{xx}(k)$ and $S_{yy}(k)$ are the diagonal elements of covariance matrix:

$$S(k) = HP(k|k-1)H^T + R \quad (15)$$

- v_2 : change in heading, defined as follows:

$$v_2(k) = \alpha_{LT}(k) - \alpha_{LT}(k-1) \quad (16)$$

where $\alpha_{LT}(k)$ and $\alpha_{LT}(k-1)$ are heading estimate computed with past data points (i.e., N equals to 3) at sampling instants k and (k-1) respectively,

$$\alpha_{LT}(k) = \mu[\sum_{i=1}^{N}(y_i - \bar{y})^2 / \sum_{i=1}^{N}(x_i - \bar{x})^2]^{1/2} \quad (17)$$

where

$$\mu = \mathrm{sgn}\sum_{i=1}^{N}(y_i - \bar{y})(x_i - \bar{x}) \quad (18)$$

$$\bar{x} = (1/N)\sum_{i=1}^{N} x_i \quad (19)$$

$$\bar{y} = (1/N)\sum_{i=1}^{N} y_i \quad (20)$$

The basis of using neural network to abstract acceleration parameter is that when there is no maneuver, the mean of innovation sequence (a group of data consists of past innovation value) equals to zero, but when there is a maneuver, it no longer equals to zero. The previous theory is based on the assumption of constant process noise and measurement noise, but it seems colliding with the purpose of adjusting the covariance of measurement noise on-line. This can almost be solved by using data from different sensors .Suppose there are several sensors, when measurement noise of partial sensors change, the acceleration parameter can still be learned in relatively more sampling periods comparing with constant measurement noise. There are many reasons for the change of measurement noise, for example, dithering of sensors, variance of environment, existence of clutter. So if we choose different kinds of sensors which have different work conditions and different sensitivities to bad weather like rain or fog, there might be only a few sensors whose measurement noise changed with other measurement noise of sensors staying invariable, in that case, neural network can still effectively abstract acceleration parameter.

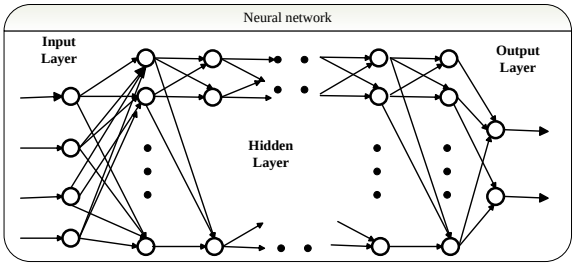


Fig. 1: Multilayer feedforward neural network architecture

2.3. Fuzzy logic system

The main advantages of using fuzzy logic system are the simplicity of the approach and the capacity of dealing with imprecise information, and also the possibility of including heuristic knowledge about phenomenon under consideration.

The ability of fuzzy logic system to adaptively adjusting the covariance of measurement noise has been proved in reference 7. The foundation of this application is named innovation-based adaptive estimation algorithm [11]. The basic idea behind this algorithm is to make the actual value of the covariance of the residual consistent with its theoretical value. The innovation sequence $r(k)$ has a theoretical covariance $S(k)$ defined in Equation 15. Given the availability of the innovation sequence $r(k)$, its actual covariance $\hat{C}_{rk}$ is approximated by its sample covariance through averaging inside a moving estimation window of size N,

$$\hat{C}_{rk} = \frac{1}{N}\sum_{i=i_0}^{N} r_i r_i^T \tag{21}$$

where $i_0 = k - N + 1$ is the first sample inside the estimation window. The window size N is chosen empirically to give some statistical smoothing.

Thus, if it is found that the actual covariance of $r(k)$ has a discrepancy with its theoretical value, then adjustments have to be made to R in order to correct this mismatch. A new variable named Degree of Matching (DoM) is defined to detect the discrepancy of $S(k)$ and its actual value $\hat{C}_{rk}$,

$$DoM_k = S_k - \hat{C}_{rk} \tag{22}$$

Then a Fuzzy Inference system can be constructed to adjust R based on rules as follows:

- If $DoM \cong 0$ (this means $S(k)$ and $\hat{C}_{rk}$ match almost perfectly) then maintain R unchanged;
- If $DoM > 0$ (this means $S(k)$ is greater than its actual value $\hat{C}_{rk}$) then decrease R;
- If $DoM < 0$ (this means $S(k)$ is smaller than its actual value $\hat{C}_{rk}$) then increase R;

R is adjusted in this way:

$$R_k = R_{k-1} + \Delta R_k \tag{23}$$

where ΔR_k is the output of the FIS.

When there is more than one sensor, the degree of confidence of sensor data can also be generated by FLO (fuzzy logic observer) to fuse the data. It takes covariance of measurement noise R_k adjusted according to Eq.23 and DoM as inputs and exports the w (degree of confidence).The rules can be set as follows:

- If $|DoM| = ZE$ and $R = ZE$, then $w = G$;
- If $|DoM| = ZE$ and $R = S$, then $w = G$;
- If $|DoM| = ZE$ and $R = L$, then $w = AV$;
- If $|DoM| = S$ and $R = ZE$, then $w = G$;
- If $|DoM| = S$ and $R = S$,then $w = AV$;
- If $|DoM| = S$ and $R = L$,then $w = P$;
- If $|DoM| = L$ and $R = ZE$,then $w = AV$;
- If $|DoM| = L$ and $R = S$, then $w = P$;
- If $|DoM| = L$ and $R = L$,then $w = P$;

where ZE means zero, S means small, L means Large, G means Good, AV means average, P means Poor.
The FIS and FLO can be integrated into one system shown as follows:

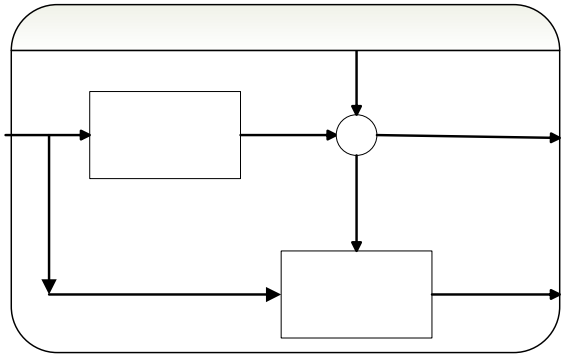


Fig.2 Fuzzy logic system.

From fig.2, we can see Fuzzy logic system takes DoM as input and provides adjusted R to Kalman filter and w (degree of confidence) for fusion.

2.4. New data fusion architecture

This new data fusion architecture is constructed to solve the following two issues: first, learning the acceleration parameter of maneuver with neural network, especially for the fast maneuver of short term acceleration; second, adjusting covariance of measurement noise on-line with the help of fuzzy logic system. The architecture is shown in figure 3.

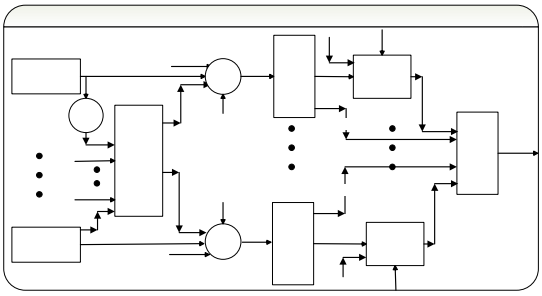


Fig. 3: New data fusion architecture

There are three main algorithms used in this architecture: neural network, Kalman filter and fuzzy logic. Their relationships have been shown in the figure 3. First, all of new measurements of sensors at time k will be transmitted into neural network, fuzzy logic systems and Kalman filters. Neural network module will export acceleration parameter to fuzzy logic systems and Kalman filters. The fuzzy logic systems will adjust R parameters (covariance of measurement noise) of the Kalman filters while offering corresponding degree of confidence of sensors to the fusion module at the same time. The fusion module will fuse the outputs of Kalman filters according to the degree of confidence.

There are several signs in figure 3, which denote different meanings in the architecture as follows:

- v_1: intensity of acceleration $v_1(k)$ defined in Eq.14;
- v_2 : change in heading $v_2(k)$,defined in Eq.16;
- v_3 and v_n : some other inputs to neural network which are used to estimate acceleration parameter, their detailed forms depend on the property of sensors.
- $Z(k)$: measurement of sensor at sampling time k;
- $S(k)$: covariance of innovation sequence, defined in Eq.15
- a : estimate acceleration parameter at sampling instants k;
- $X(k-1)$: estimate value at sampling instants k-1;
- DoM : Degree of Matching, input of fuzzy logic system, defined in Eq.22;
- $R(k)$: covariance of measurement noise at sampling instants k, one of the outputs of the fuzzy logic system.
- w: degree of confidence of a sensor, one of the two outputs of the fuzzy logic system..

In this architecture, there are two modules need to be simply introduced: math operation module and fusion module. Math operation module is used to transfer inputs to outputs after simple mathematic operation. And fusion module is used to fuse outputs of Kalman filters to a value which is more close to the real value based on fusion algorithm as follows:

$$\tilde{X}(k) = \frac{\sum_{i=1}^{m} w_i \hat{x}_i(k)}{\sum_{i=1}^{m} w_i} \tag{24}$$

where m is the number of sensors, w_i is the degree of confidence of i^{th} sensor, $\hat{x}_i(k)$ is the estimation of real value at sampling instants k offered by i^{th} Kalman filter, $\tilde{X}(k)$ is the fusion result at sampling instants k.

3. Simulation and performance evaluation of the architecture

In order to demonstrate the effectiveness of this architecture, a simulation is presented in this section.

Suppose there are two sensors in our tracking system, both of them provide two dimensions measurement data. They have different work conditions and sensitivities to bad whether. The tracking model is linear as follows:

$$\begin{bmatrix} x(k) \\ y(k) \\ v_x(k) \\ v_y(k) \end{bmatrix} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x(k-1) \\ y(k-1) \\ v_x(k-1) \\ v_y(k-1) \end{bmatrix} + \begin{bmatrix} \frac{1}{2}T^2 & 0 \\ 0 & \frac{1}{2}T^2 \\ T & 0 \\ 0 & T \end{bmatrix} \begin{bmatrix} u_x \\ u_y \end{bmatrix} + \begin{bmatrix} w_x \\ w_y \\ w_{v_x} \\ w_{v_y} \end{bmatrix} \tag{25}$$

$$\begin{bmatrix} z_x(k) \\ z_y(k) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ y(k) \\ v_x(k) \\ v_y(k) \end{bmatrix} + \begin{bmatrix} v_x \\ v_y \end{bmatrix} \tag{26}$$

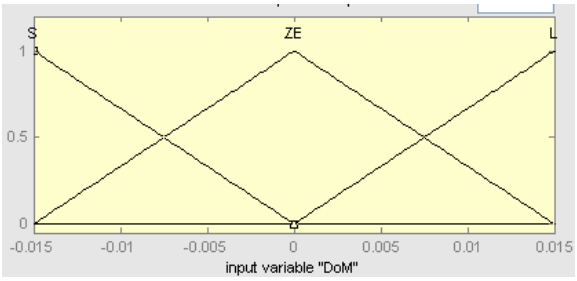
where T is sampling time.

Neural network has four inputs and two outputs. The number of neutron in the hidden layer can only be decided after training. In order to have a good accuracy of estimating acceleration, we created 2000 training data by simulation as following design:

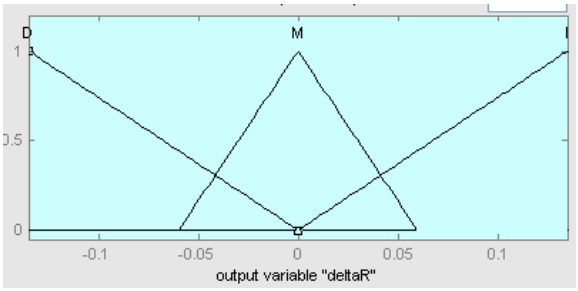
- Acceleration: range from 0 to 20, divided into 20 levels;
- Initial velocity: range from 100 to 200, divided into 10 levels.
- Covariance of measurement noise: range from 0.1 to 1, divided into 10 levels.

The training result shows 35 neurons are enough for the hidden layer.

Fuzzy logic system consists of two parts: FIS and FLO. The input and output of FIS are set as fig.4. The input and output of FLO are set as fig.5.

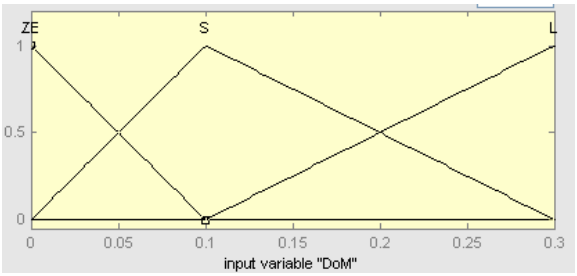


(a)

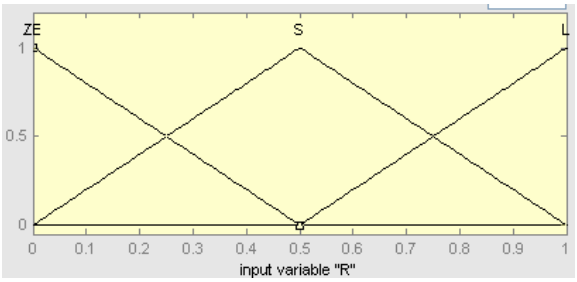


(b)

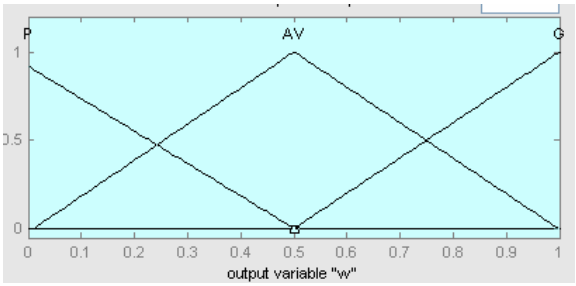
Fig.4: (a) Membership function for input variable DoM (b) Member function for output variable ΔR .



(a)



(b)



(c)

Fig.5: (a) Membership function for input variable DoM; (b) Membership function for input variable R; (c) Membership function for output variable w.

There are two questions we need to be solved. First, whether can this architecture work effectively when short term maneuver and variety of measurement noise happened orderly; Second, if covariance of measurement noise of both sensors change at the same time, whether can this architecture work correctly.

Two tests were outlined for the sake of answering the previous two questions.

3.1. Test one

Suppose a target is moving with the initial velocity of 100m/s, sampling period of target tracking system is 4s, covariance of measurement noise of both sensors is 0.1, a short term maneuver happens at sampling instants 5. This maneuver consists of sharp acceleration of 3 m/s^2 and lasting for only one sampling time. Then covariance of measurement noise of one sensor changes from 0.1 to 0.5 at sampling instants 10 and kept unchanged in the following sampling time. Simulation result of x coordinate position error (y coordinate position error is omitted because x and y coordinate have the same simulation conditions) is shown in fig.6 where performance of the new fusion architecture is compared with an architecture with neural network only and an architecture with fuzzy logic system only. The result shows this new fusion architecture can adjust acceleration parameter in nearly one sampling period and adaptively change the covariance of measurement noise in several sampling periods. On the contrary, the architecture with neural network only and the one with fuzzy logic system only lost targets when variety of measurement noise and maneuver happens respectively.

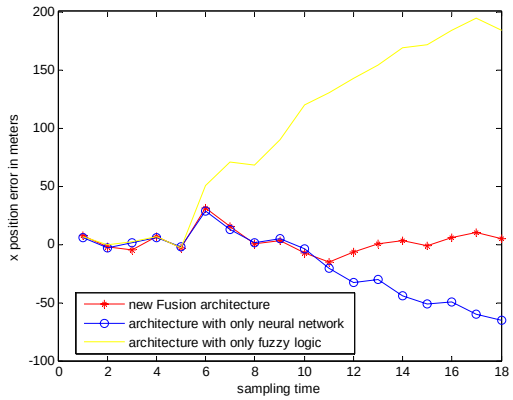


Fig.6: x coordinate position error of tracking system using three different fusion architectures.

3.2. Test two

Suppose a target is moving with the initial velocity of 100 m/s, sampling period of target tracking system is 4s, covariance of measurement noise of both sensors is 0.1. At sampling instants 5, covariance of measurement noise of both sensors changes from 0.1 to 0.5. On the purpose of comparison, we attain another set of data with only one sensor's measurement noise changing at sampling instants 5 from 0.1 to 0.5. Simulation result of x coordinate position error is shown in fig.7 which brings a bad news, that when the measurement noise of both two sensors change together, this architecture can not work correctly. In order to avoid this situation, we need to carefully choose heterogenous sensors which have different working mechanics and sensitivities to bad weather and also install the sensors on different platforms.

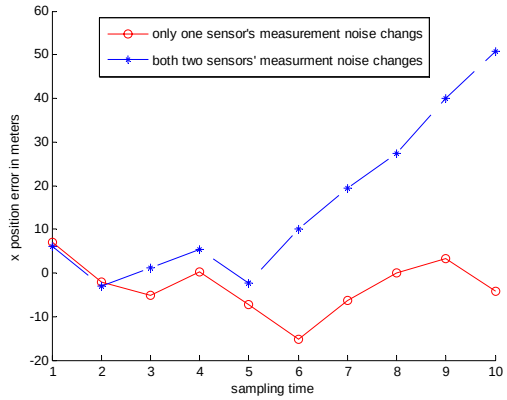


Fig.7: x coordinates position error of tracking system using this architecture.

4. Conclusions

The main purpose of this paper is to present a new fusion architecture used in target tracking system which can solve the problem of maneuver of short term acceleration and adaptively adjust the covariance of measurement noise of sensors. The simulation results show this architecture can solve previous two issues, but also has the defection that sensors used in the tracking system need to be chosen carefully to prevent variety of measurement noise of all the sensors from happening at the same time. The sensors should have different working mechanics and sensitivities to bad weather and install the sensors on different platforms. So our next study is to find some algorithm to solve that problem.

Acknowledgement

This work is partially supported by National Nature Science Foundation of China (Grant No. 60572079).

References

[1] H. Blom, An efficient filter for abruptly changing systems, *IEEE Proceedings of 23rd Conference on Decision and Control*, 23:656-658, 1984.

[2] Y.T. Chan, A.T.C. Hun, A.G.C. Hu and J.B. Plant, A Kalman Filter Based Tracking Scheme with Input Estimation, *Transaction on Aerospace and Electronic Systems*, 15:237-244, 1979.

[3] M.K. Sundareshan, F. Amoozegar, Neural network fusion capabilities for efficient implementation of tracking algorithms, *Society of Photo-Optical Instrumentation Engineers*, 36: 625-965, 1997.

[4] M.W. Owen and A.R. Stubberud, NEKF IMM Tracking Algorithm, *Signal and Data Processing of Small Targets*, 5204:223-233, 2003.

[5] J.G. Lin, The adaptation of observation noise covariances and adaptive Kalman filtering, *Decision and Control*, 12:366-370, 1973.

[6] J. Cao, Jianchenga Fang and Weia Sheng, Neural Adaptive Kalman Filter's Application in MIMU GPS MMC, *Sixth International Symposium on Instrumentation and Control Technology: Signal Analysis, Measurement Theory, Photo-Electronic Technology, and Artificial Intelligence*, 6357:63574V1-63574V7, 2006.

[7] P.J. Escamilla-Ambrosio, N. Mort, A hybrid Kalman filter-fuzzy logic architecture for multisensor data fusion, *International Symposium on Intelligent Control*, pp. 364-369, 2001.

[8] P. S. Maybeck, Stochastic models estimation and control, *Academic Press*, New York, 1979.
[9] K. Hornik, M. Strinchcombe and H. White, Multilayer feedforward networks are universal approximators, *Neural Net*, 2:359-366,1989.
[10] K. Funahashi, On the approximate realization of continuous mappings by neural networks, *Neural Net*, 2:183-192, 1989.
[11] A. H. Mohamed and K. P. Shwarz, Adaptive Kalman filter for INS/GPS, *Automatic Control*, 16:193-203, 1999.