A particle filter and SVM integration framework for fault-proneness prediction in robot dead reckoning system

Lingli Yu, Min Wu, Zixing Cai, Yu Cao
School of Information Science and Engineering
Central South University
Changsha, Hunan, 410083
CHINA
llyu@csu.edu.cn

Abstract: - This paper proposes an integrated framework for fault prediction in the robot dead reckoning system. The integrated framework is built by particle filter and support vector machine (SVM). On the basis, the weighted fault probability parameters can be extracted to train the prediction model. Different from the traditional particle filter fault prediction model, the proposed framework can overcome difficulties of the empirical threshold setting for decision-making. On the other hand, particle filter can not only estimate the system state values, but also obtain the residual errors that are yielded by comparing with the actual measured values. The average relative error is calculated to reduce its computing complexity in fault prediction process. Furthermore, an improved particle filter combined with support vector machine (PF-SVM) integration framework for fault-proneness prediction was devised in robot dead reckoning system. This framework estimates the particle filter process according to system state values and observed parameters to train SVM model. Finally the simulation experiments demonstrate that the PF-SVM integration framework can increase computing efficiency for fault-proneness prediction, and keep relatively high prediction accuracy at the same time. Besides that, the corresponding time step of malfunction can be precisely predicted.
Key-Words: - Particle filter, support vector machine, fault prediction, weighted fault probability, integration framework, dead reckoning system

1 Introduction

Fault-proneness prediction mainly deals with faults that will happen according to the past and current states of the system [1], and it is a natural extension to the problem of fault detection and identification (FDI). With the developing demand for a higher operational efficiency and security reliability in the mobile robot systems, fault-proneness prediction has become a key issue world-wide. A more critical issue in the study of fault diagnosis was raised up in the field of hybrid system fault-proneness prediction. The robot dead reckoning system is such a kind of classic hybrid system that has multi-variables of both discrete and continuous values of system states. Unfortunately, in the robot dead reckoning system, fault-proneness cannot be easily measured, because the directly collected sensor data cannot distinctly attribute to the fault characteristics, so it is difficult to identify the fault symptoms. However, it can be obtained on the basis of extracted characteristic parameters indirectly by statistical and quantitative descriptions of system states.

The existing methods of fault-proneness prediction are generally classified into three major types: the model-based fault-proneness prediction, including Recursive Least Square (RLS), Kalman filter and Particle filter methods [2]; and the knowledge-based fault-proneness prediction methods such as expert system [3] and fuzzy logic technology, which are not a closed and clear-cut discipline. It incorporates an emerging family of problem-stating and problem-solving methods that attempt to mimic natural intelligence [4]; and the data-based fault-proneness prediction, for instance, time sequence analysis, grey theory and some machine learning methods [5]. Particle filter, a sequential Monte-Carlo technique, which is widely applied to the State Space Model, is one of the classical model-based fault-proneness prediction methods. Particle Filter is able to handle discrete states and continuous states simultaneously, and allows the measured data to join from multiple sources subjecting to the constrained conditions. Up to now, several applications of Particle Filter for fault-proneness prediction have been devised [6-8] and further studied [9]; the prediction reliability of the system was computed and measured through the Monte Carlo simulation based on the results of particle filtering and fault-proneness prediction. Furthermore, the concept of strong tracking particle was put forward by introducing a strong tracking method into the particle to resolve the problems of

particle degeneracy and its poor ability to track saltatory states [2]. Though a great amount of work have been done to deal with the hybrid system for fault-proneness prediction using Particle Filter, they avoided the decision process for prediction. Most of the research regarded the weighted fault probability as significant characteristic parameters and tried to set an empirical threshold for decision-making to predict whether there will be a fault mode. But it's difficult to select an adaptive threshold for different kinds of system. Fortunately, this can be solved by applying machine learning technologies to decision process of fault-proneness prediction. Unlike some traditional models for fault-proneness prediction, machine learning proposed in [10] confirmed the superior performance of Support Vector Machines over Artificial Neural Networks when treating fault-proneness prediction as a binary classification task. Some researchers have made great progress on Support Vector Machine (SVM) for fault-proneness prediction; we can find its great worth from [11-13]. In this article, we focus to build a fault-proneness prediction model by SVM for the special robot dead reckoning system, where several particle filter and Support Vector Machine integration frameworks are devised and compared for fault-proneness prediction (a model-based and a data-based particle filter). These frameworks extract the weighted fault probability and the characteristic parameters of system states in the process of particle filtering in order to train SVM and adapt its chosen threshold, thus improve the runtime efficiency of algorithms and keep higher accuracy rates of fault prediction. Above all, we have demonstrated that one of these algorithms can accurately predict the certain time step of fault with a relatively high efficiency.

The rest of this paper is organized as follows. In Section 2, some preliminaries about particle filter and Support Vector Machine are briefly reviewed. Particle filter based fault-proneness prediction and corresponding fault models devised for the robot dead reckoning system are presented in Section 3. Particle filter and SVM integration framework for fault-proneness prediction are described in Section 4 in details; including the utilization of the weighted fault probability in the SVM based Particle filter for fault-proneness prediction; the residual error obtained to improve the particle filter based fault-proneness prediction; and particle filter and Support Vector Machine integration framework based fault-proneness prediction. Simulation experiment results on the platform of MORCS-1 robot dead reckoning system for fault-proneness prediction are analyzed in Section 5, comparing effectiveness and efficiency of each devised algorithm for fault-proneness prediction. Finally, we draw our conclusions and the future development in section 6.

2 Preliminaries

2.1 Particle filter

Particle Filter is a kind of nonlinear filtering algorithm on the basis of the sequential Monte Calo simulation method, which is a posterior density function of recursive approximation of states, and it converges to the real posterior density as the particle population tends to infinity. $X = \{X_t , t \in N\}$ is a set of values belonging to R^{n_x}, set the initial distribution as $p(x_0)$ and $p(x_t \mid x_{t-1})$ as the transition probability, which is also defined in the equation (1), $\{\omega\}_{t\geq 0}$ is a sequence of independent random variables.

$$x_t = f_t(x_{t-1}, \omega_t) \tag{1}$$

Meanwhile, the independent measurement of noise is $Y = \{Y_t, t \in N\}$. Equation (2) defines the marginal distribution $p(y_t \mid x_t)$, in which $\{\upsilon_t\}_{t\geq 0}$ is a sequence of independent random variables, but may not be the Gaussian noise.

$$y_t = g_t(x_t, \upsilon_t) \tag{2}$$

where $x_{0:t} = \{x_0, \cdots, x_t\}$ and $y_{1:t} = \{y_1, \cdots, y_t\}$ are respectively the characteristic parameters of system states and the observed values in moment t , which are utilized for estimating the posterior distribution $p(x_{0:t} \mid y_{1:t})$ and marginal distribution $p(x_t \mid y_{1:t})$ [14]. This task can be implemented by two continuous procedures: predicting and filtering [15].

On the one hand, the prediction of the next time can use both the previous knowledge of state estimation and the process model to generate the priori state probability of density estimation, as shown in equation (3).

$$p(x_{0:t} \mid y_{1:t-1}) = \int p(x_t \mid x_{t-1}) p(x_{0:t-1} \mid y_{1:t-1}) dx_{0:t-1} \tag{3}$$

On the other hand, the posterior probability density function generated after the filtering step is shown in equation (4):

$$p(x_{0:t} \mid y_t) \propto p(y_t \mid x_t) \cdot p(x_t \mid x_{0:t-1}) \cdot p(x_{0:t-1} \mid y_{1:t-1}) \tag{4}$$

After the process of resample, the particle swarm $\{\tilde{x}_{0:t}^{(i)}\}_{i=1,\ldots,N}$ is updated as independent and identical

experience distribution samples by resetting its weights as $\tilde{w}_t^{(j)} = N^{-1}$ as shown in equation (5).

$$\tilde{\pi}_t^N(x_{0:t}) = \frac{1}{N}\sum_{i=1}^{N} N_t^{(i)} \delta\left(x_{0:t} - \tilde{x}_{0:t}^{(i)}\right) = \frac{1}{N}\sum_{i=1}^{N} \delta\left(x_{0:t} - \tilde{x}_{0:t}^{(i)}\right) \quad (5)$$

Here, $\delta(\bullet)$ is Dirac delta function, and it's easy to prove that equation (5) approaches to the true posterior distribution when $N \to \infty$.

2.2 Support vector machine

SVM is on the basis of statistical theory of VC dimension and structural risk minimization inductive principle. With limited sources of samples, Support Vector Machine hunts the best results of both complexity of modeling and learning ability in the expectation of getting optimal generalization ability. The most straightforward expression of structural risk is to construct classifier and utilize the difference between the sort results of sample data and the true ones. Obviously, the higher the VC dimension is, the worse the generalization capacity is, with increasing computing complexity. And Support Vector Machine is a kind of structural risk minimization algorithm.

2.1.1 Linear classifier

Each sample of Support Vector Machine contains targeted classification value (or tag type) and some properties (or characteristic parameters).Considering the linearly separable cases, given the training sample $\{x_i, y_i\}_{i=1}^N$, $i = 1,...,N$, in which $x_i \in R^n$ is the *i*th sample of input model, $y \in \{1,-1\}^l$ is the set of classification tag for training. According to the principle of Support Vector Machine, hunting the optimal classification hyperplane is equal to seeking the maximal margin between positive and negative classes of separation. Support vectors are regarded as the closest ones to the decision-making boundary shown in Fig.1, and x_1, x_2 satisfied equation (6).

$$\begin{cases} w \cdot x_1 + b = 1 \\ w \cdot x_2 + b = -1 \end{cases} \quad (6)$$

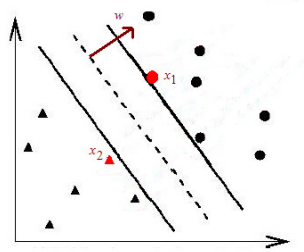


Fig.1 Simple binary classifier

The hyperplane corresponding to $w\boldsymbol{x} + b = 0$ is the decision boundary. Here w is an adjustable weight vector, and is also a normal vector of the hyperplane, and b is the offset, or the constant term of the hyperplane. So the margin is:

$$dis = \frac{w}{\|w\|} \cdot (x_1 - x_2) = \frac{2}{\|w\|} \quad (7)$$

In equation (7), the task is to maximize the margin, looking for $\min\|w\|$, or $\min\frac{\|w\|^2}{2}$ in other words. For any (x_i, y_i), there exists:

$$\begin{cases} w \cdot x_i + b \le 1, y_i = -1 \\ w \cdot x_i + b \ge -1, y_i = 1 \end{cases} \quad (8)$$

Combining equation (7) and (8) to seek the optimal classification hyperplane is equivalent to find the maximal margin between the positive and negative classes of separation. By pulling into the Lagrange multiplier, we settle the issues above and obtain

$$w_0 = \sum_{i=1}^{N} a_i^* y_i x_i, \quad b_0 = 1 - w_0 x^{(s)}, y^{(s)} = 1,$$

to solve the problem of linear classifier.

2.2.2. Kernel function

Kernel function, a convenient inner product function to transform the input values from low dimension space to high dimension space, makes it easy to compute and reduce VC dimension. Different kernel functions applied to learning machines can construct different types of nonlinear decision hyperplanes, from which we will obtain different algorithms of Support Vectors. Considering the nonlinear case, $\boldsymbol{x}$ is sufficiently mapped into a high dimension space as shown in equation (9) and (10).

$$\min_{w,b,\xi} \frac{1}{2} w^T w + C\sum_{i=1}^{l} \xi_i \quad (9)$$

Subject to
$$y_i(w^T \phi(x_i) + b) \ge 1 - \xi_i, \quad \xi_i \ge 0. \quad (10)$$

in which $\boldsymbol{C}$>0 is a penalty parameter of fault, and $K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_i)$ is called kernel function.

3 Robot dead reckoning system fault model and fault prediction based on particle filter

3.1 Fault prediction and fault space model for dead reckoning

The expression of hybrid dynamic system (HDS) is shown in equation (11); $s_t \in \mathbb{S}$ is the discrete system mode at moment t where $\mathbb{S}$ is the finite set

of discrete system modes; $x_t \in R^{n_x}$ is the continuous state variable of system at moment t; $z_t \in R^{n_z}$ is the measurement of the system at moment t; $u_t \in R^{n_u}$ is the input of dead reckoning system at moment t; $\boldsymbol{v}_t$ and $\boldsymbol{n}_t$ are considered as the process noise and measurement noise respectively.

$$\begin{cases} \dot{x}_t = f(s_t, x_t, u_t) + v_t \\ z_t = h(s_t, x_t, u_t) + n_t \end{cases} \tag{11}$$

As we all know that fault-proneness prediction in the mobile robot dead reckoning system is a typical HDS, which can be described as following: given its system model, on the basis of observation sequence $y_{0:t} = \{y_0, y_1, \ldots, y_t\}$, continuous control input sequence $u_{0:t} = \{u_0, u_1, \ldots, u_t\}$ and the continuous state sequence $x_{0:t-1} = \{x_0, x_1, \ldots, x_{t-1}\}$, the fault-proneness prediction estimate the state x_t at moment t and obtain the discrete state s_{t+k} at moment $t+k$. Setting $x = [\omega_L, \omega_R, \omega]^T$ as state variables and $u = [u_L, u_R]^T$ as input vector, where u_L and u_R are the input control variables that respectively signify the set speed of the left wheel and right wheel, and $y = [\omega_L, \omega_R, \omega]^T$ as observation vector; and the following $p(x_t \mid x_{t-1}, s_t)$ is expressed in equation (12) in linear Gaussian model:

$$x_t = A(s_t)x_{t-1} + B(s_t)\omega_t + F(s_t)u_t \tag{12}$$

and $p(y_t \mid x_t, s_t)$ is expressed as shown in equation (13) in linear Gaussian model:

$$y_t = C(s_t)x_t + D(s_t)v_t + G(s_t)u_t \tag{13}$$

where $s_t \in S = \{S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7\}$; $\boldsymbol{S}_0$ represents the normal working state of the system; $\boldsymbol{S}_1$ represents malfunction with its left wheel encoder; $\boldsymbol{S}_2$ represents its losing efficiency with the right wheel encoder; $\boldsymbol{S}_3$ represents the failure with its gyroscope; $\boldsymbol{S}_4$ represents that both wheel encoders lose their efficiency; $\boldsymbol{S}_5$ represents the loss of left wheel encoder and gyroscope; $\boldsymbol{S}_6$ represents the fault of right wheel encoder and gyroscope; $\boldsymbol{S}_7$ represents the malfunction of all the three at the same time. The system parameter matrixes are:

$$A(s) = 0; B(s) = 0; G(s) = 0;$$

$$F(s) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ c & d \end{bmatrix}; D(s) = \begin{bmatrix} e & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & g \end{bmatrix}$$

where $c = -r_L / \omega, d = r_R / \omega; e = 0.1; f = 0.1;$ g= 0.0223. And e, f, g respectively represents standard deviation of noise of the left wheel, right wheel, and gyroscope. The motion patterns of the mobile robot subject to the detectable fault set of system state of a certain sensor. For example, it is difficult to confirm whether there will be a fault occurring with gyro sensor when the outputs of robot is still the encoders; similarly, the fault of gyro is hard to figure when the robot is in straight line mode (M2). Regarding to the differences among the kinematic models of different fault modes, we use different observation equations to express, for example:

$$C(s_0) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}; C(s_1) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix};$$

$$C(s_2) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}; \cdots C(s_7) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

3.2 The experimental platform MORCS-1

Mobile robots are used in active service for the assisted living of elderly people [16]. We use the mobile robot (MORCS-1) of Central South University as the experiment platform and the picture of MORCS-1 is shown in Fig.2, whose dead reckoning system is composed of encoders for both left and right wheels, and mobile-robot gyroscope. The encoders measure the angular velocity of left and right driving wheels respectively; meanwhile the gyro measures the yaw rate of the robot. As the tuning is obtained by the difference of their speeds between left motor and right motor in real-time, and the motion pattern is decided by the speed of the left and/or the right driving wheel. The robot may in the static mode (M1, $u_t^L = u_t^R = 0$), the straight line mode (M2, $u_t^L = u_t^R \neq 0$), the rotation revolved around the left wheel (M3, $u_t^L = 0$, $u_t^R \neq 0$), the rotation about the right wheel (M4, $u_t^L \neq 0, u_t^R = 0$), and other rotations (M5, $u_t^L \neq 0, u_t^R \neq 0, u_t^L \neq u_t^R$).

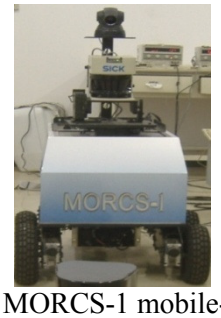


Fig.2 MORCS-1 mobile-robot

3.3 Fault prediction based on particle filter

Suppose x_t follows the first-order Markov process in mode s_t, and the measurement sequence y_t is mutually independent. Set the prior distribution of original state x_1 as $p(x_o)$, and the prior distribution of mode s_t as $p(s_o)$.

Step1. estimate the current state using $p(s_t \mid y_{1:t})$

(1) Initializaiton

The initialized time is $t=0$, sampling for samples $i=1,2,...,N$: $d_0^i \sim p(d_0)$, $x_0^i \sim p(x_0 \mid d_0)$, set the initial weight $\omega_0^i = 1/N$, and then obtain its equal weighted sample set $\{d_0^i, x_0^i, 1/N\}_{i=1}^N$.

(2) Update the weighted values

Calculate $\{s_t^i, x_t^i\}_{i=1}^N$ according to $p(s_t \mid s_{t-1}^i)$, and compare the observed value with prediction result of the output of particle filter to calculate the weighted values of importance.

$$\omega_t^i = \omega_{t-1}^i \frac{p(y_t \mid s_t^i) p(s_t^i \mid s_{t-1}^i)}{q(s_t^i \mid s_{t-1}^i, y_t)}$$ for each sample.

(3) Weights Normalization $\tilde{\omega}_t^i = \dfrac{\omega_t^i}{\sum_{i=1}^N \omega_t^i}$.

(4) State estimation

Given the set of $\{d_t^i, x_t^i, \tilde{\omega}_t^i\}_{i=1}^N$ to approximately estimate $p(s_t \mid y_{1:t})$:

$$p(s_t \mid y_{1:t}) = \sum_{j=1}^N \sum_{i=1}^N \tilde{\omega}_t^i \delta_{s_t^i}(s_t^i - s_t^j) \qquad (14)$$

Step2. estimate the posterior density distribution $p(s_{t+k} \mid y_{1:t})$ at moment *t*.

$$\begin{aligned} p(s_{t+k} \in S \mid y_{1:t}) &= \int_S p(x_{t+k} \mid y_{1:t}) ds_{t+k}, k \geq 2 \\ &= \int p(s_t \mid y_{1:t}) [\prod_{j=t+1}^{t+k} p(s_j \mid s_{j-1})] ds_{t:t+k-1} \end{aligned} \qquad (15)$$

Let equation (14) into equation (15):

$$p(s_{t+k} \in S \mid y_{1:t}) = \sum_{i=1}^N \omega_t^i \int p(s_{t+1} \mid s_t^i) \prod_{j=t+2}^{t+k} p(s_j \mid s_{j-1}) ds_{t+1:t+k-1}$$

Here $ds_{t+1:t+k-1} = ds_t \cdot ds_{t-1} \cdots ds_{t+k+1}$, the prediction of malfunction after k steps is based on information at moment t. As a result, we can utilize the posterior density distribution of state $p(s_t \mid y_{1:t})$ at moment t to estimate state $p(s_{t+k} \mid y_{1:t})$ at moment $t+k$, as shown in Table 1.

Table 1 the pseudo code of fault prediction based on particle filter

```
(1) Sampling
    for j=1 to k
        for i=1:N
            sample s^i_{t+j} ~ p(s_{t+j} | s_{s+j-1})
            and s^i_{0:t+j} : (s^i_{0:t+j-1}, s^i_{t+j})
        end
    end
    Obtain the samples {s^i_{1:t+k}; i = 1,2,...,N}
(2) State prediction p(s_{t+k} | y_{1:t}) = Σ_{i=1}^N ω̃^i_t δ_{s_{t+k}}(s_{t+k})
(3) Decision-making of predicting fault probability
```

3.4 Weighted fault probability of particle filter for decision-making of fault-proneness prediction

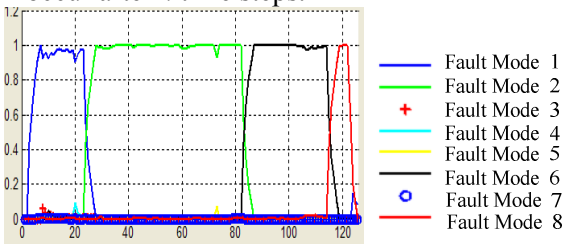
The prediction of fault-proneness probability $p(s_{t+k} \mid y_{1:t})$ at moment $t+k$ is decided by several statistical indicators: fault-proneness probability and its mathematical expectation or variance. In paper, weighted fault probability was utilized for prediction of fault-proneness occurring probability [17]. Weighted fault probability, one of the characteristic parameters for the judgment of malfunction, is an efficient approach to predict fault occurring. With *k* steps ahead of time, the weighted fault probability depends on its forecasting from moment *t* to *t+k*, whose calculation is shown in equation (16).

$$\Pr ob(s_{t+k}) = \sum_{i=j}^N \Pr ob(s_{t+k-i}, i) \cdot w_i \qquad (16)$$

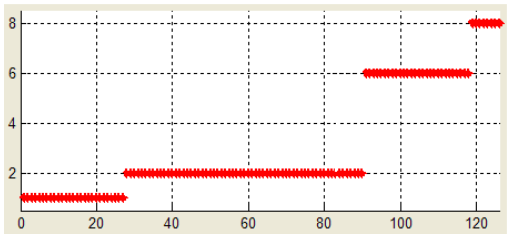
While $1 \leq j \leq N$, ***N*** represents the maximum allowed size of decision-making step for prediction at moment *t+k* (for example *N*=5). $\Pr ob(s_{t+k})$ represents the prediction result of weighted fault probability whose step size is *j* at moment *t+k* in equation (16); $\Pr ob(s_{t+k-i}, i)$ shows the prediction of weighted fault probability at current moment $t+k-i$ in which w_i is the corresponding particle weight. The weighted fault probability reflects the accumulation process of fault degree. As a result, the weighted fault probability can record variation tendency of system faults, which is an important parameter in decision-making of fault prediction.

We utilize weighted fault probability based particle filter to predict fault-proneness of the robot dead reckoning system in MORCS-1, whose results are shown in Fig.3 (a) by calculating the weighted fault probability respectively, and in Fig.3 (b), the thick line represents corresponding real-time fault mode. After analytical comparison, fault will occur when its corresponding weighted fault probability verges

on 1. If there is tendency of fault, its weighted fault probability will be significantly rising. Therefore, it is important to set an appropriate experience-based threshold of the weighted fault probability to predict the impending fault probability. If the weighted fault probability of a certain fault mode is greater than the setting threshold, or the sum of other weighted fault probabilities is much less than another threshold, we can safely draw the conclusion that this fault mode will occur after k time steps.



(a)



(b)

Fig.3 Comparison of weighted fault probability and the real fault mode

4 Several proposed fault-proneness prediction methods based on particle filter and support vector machine (PF-SVM) integration framework

4.1 Decision-making of fault-proneness prediction utilizing weighted fault probability of particle filter on the basis of support vector machine

It is practical to complete fault prediction using threshold of weighted fault probability. However, the threshold selection mainly depends on experience. It is not ideal when it comes to the calculation of the weighted fault probability; or the inappropriately selected threshold based on experience cannot be adjusted adaptively, and prediction results will not be optimistic. Thus, we use the prediction model of SVM to improve the threshold setting problem.

The input characteristic parameter of Fault-proneness prediction method is the model of weighted fault probability, and output tags are prediction results of fault mode. The weighted fault probability is a process variable calculated from the estimation process of particle filtering, from which we obtain and form plenty of the training samples. Meanwhile, we adopt all kinds of the chosen fault modes to train Support Vector Machine and build the decision-making model.

Tab. 2 Fault modes list

(L, R respectively denotes the left wheel encoder and the right wheel encoder, G(Gyro) denotes the gyroscope.)

Fault Mode	Failure of components	Fault Mode	Failure of components
1	Normal	5	LR
2	L	6	LG
3	R	7	RG
4	G	8	LRG

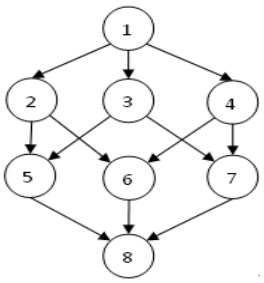


Fig. 4 Relationships among fault modes

The fault modes of the robot dead reckoning system in MORCS-1 are presented in Fig.4, in which 1~8 respectively represent the eight kinds of fault modes in table 2. The entire possible fault modes and their variation relationship can be one of them or a subset in the robot dead reckoning system. Expanding the fault relationships in Fig.4, we obtain the relation sequences of all the fault modes as shown in Fig.5. In these relation sequences, we can find that there is some redundant information, taking mode 1 & 8 for example. But it does not affect our sample training because the real-time fault occurring only belongs to a subset or a part of the test sample. And the model of Support Vector Machine can predict its result as well because the test sample has contained all of the transformation sequences among all the fault modes.

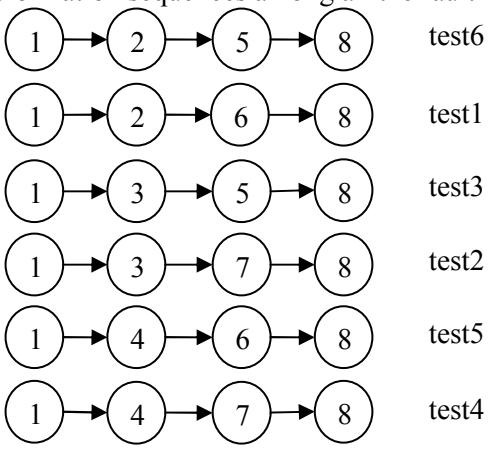


Fig.5 The set of fault relationships among test samples

Training sample is extracted from the weighted fault probability from particle filter. The randomness of "particles" is proved when we run the same set of training sample; it will generate different particles so that we will respectively obtain different values of characteristic parameters of the weighted fault probabilities each time. Nevertheless, the weighted fault probabilities are in the same trend. Hence, we can obtain abundant sets of unified characteristics as training samples.

1. Steps for the fault prediction method on the basis of weighted fault probability of Particle Filter and Support Vector Machine:

1) Extract characteristic parameters of the weighted fault probability, store samples of data according to LIBSVM software package [18], and then adjust these data in proportion;

2) Choose RBF kernel function;

3) Adopt the cross validation to select the optimal parameters *c* and *g* as shown in table 3.

Table 3 the pseudo code for parameters *c* and *g*:

```
bestcv = 0; % set the best model firstly
for log2c = -5:5,
    for log2g = -5:5,
       cmd = ['-v 5 -c ', num2str(2^log2c), ' -g ',
       num2str(2^log2g)]; %Parameters format
      cv=svmtrain(train_f_labels,train_f, cmd);
      %Training c and g between -5 and 5 separately.
     if (cv >= bestcv),
      %%Solve c and g in the best model
     bestcv = cv;
     bestc = 2^log2c;
     bestg = 2^log2g;
      end
   end
end
cmd = ['-c ', num2str(bestc), ' -g ', num2str(bestg)];
% gain the optimal parameters c and g for training model
```

4) Use the selected optimal parameters *c* and *g* to train the entire set and obtain the model of Support Vector Machine as shown in table 4.

Table 4 the pseudo code for obtaining SVM model

```
cmd = ['-c ', num2str(bestc), ' -g ', num2str(bestg)];
model = svmtrain(train_f_labels, train_f, cmd);
%model training, the "model" we obtain is a
 %syntagm data.
```

5) Test for fault-proneness prediction by the devised model mentioned above.

4.2 Residual error to improve computational complexity for fault-proneness prediction based on particle filter

The utilization of residual error to improve fault-proneness prediction with particle filter draws its lessons from the approach of fault diagnosis. Firstly the tendency of system state is estimated; then the system makes decision of pre-warning according to the average relative error.

Since computational complexity of time to extract the weighted fault probability is relatively high, there is no need to use residual error to extract and predict fault probability. And fault prediction has superiority in consideration of its time complexity, whose main flow scheme is shown in Fig.6. In the first place, we utilize Particle Filter to estimate the parameters of system states at moment t on the basis of measured value at moment *t*-1; then we compare the real-time measured value with its estimated one at moment *t*. If average relative error were greater than the setting threshold, it will be explained that there is a tendency of malfunction or the real-time measured value of variation is abnormal. That is to say, a fault of the system is just around the corner as shown in Fig.6. At moment *t*-1, the system works normal, but at moment *t* it is warned of fault and a pre-warning should be made at moment *t*-1.

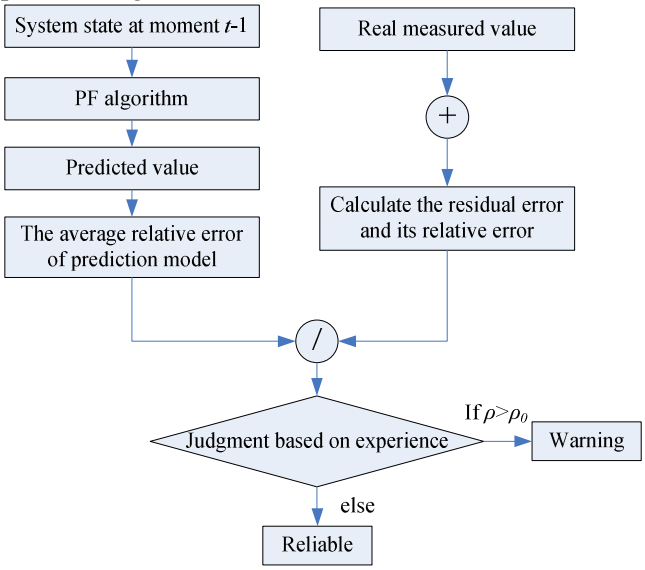


Fig. 6 Improve fault prediction with particle filter using residual error

Setting a threshold of residual error to improve fault prediction method: if the tendency of malfunction is obvious, then pre-warning will be in time, or if the tendency of fault varies gently, its pre-warning will not be made timely. The steps of its realization are listed as following:

(1)Initialization: sampling for $i=1,2,...,N$ at moment $t=0$, $d_0^i \sim p(d_0), x_0^i \sim p(x_0 \mid d_0)$ set the initial value $\omega_0^i = 1/N$, we obtain its equivalent weighted set of sample $\{d_0^i, x_0^i, 1/N\}_{i=1}^N$.

(2)Update weights: calculate $\{d_t^i, x_t^i\}_{i=1}^N$ according to $p(s_t \mid s_{t-1}^i)$, then compare each output of particle of fault prediction with the measured value, update the weight ω_t^i of importance for each sample.

(3) Weight normalization:

$$\tilde{\omega}_t^i = \frac{\omega_t^i}{\sum_{i=1}^N \omega_t^i}.$$

(4)State estimation: Given the set $\{d_t^i, x_t^i, \tilde{\omega}_t^i\}_{i=1}^N$, use $p(s_t \mid y_{1:t})$ to calculate the edge density distribution $p(d_t \mid y_{1:y})$:

(5) Decision-making and fault prediction: compare observation vector $\hat{y}_t$ with the real-time measured vector y_t and calculate their residual error:

$$p(s_t \mid y_{1:y}) = \sum_{j=1}^N \sum_{i=1}^N \tilde{\omega}_t^i \delta_{s_t^i}(s_t^i - s_t^j) \tag{17}$$

Here t is the number of prediction sample. And we obtain the relative error of residual $\tilde{e}_t$

$$e_t = \hat{y}_t - y_t \tag{18}$$

Then the ratio of prediction model between relative error and average relative error is calculated:

$$\tilde{e}_t = \left| \frac{e_t}{\hat{y}_t} \right| \tag{19}$$

and

$$\rho = \tilde{e}_t / \tilde{e} \tag{20}$$

($\rho > a$) setting the threshold value, we conclude that the malfunction will sometimes occur (the setting threshold value can be ∞).

From the discussion above, we sum up as following: the fault prediction ability of time steps depends on variation tendency of fault data such as the residual error of particle filter. If the tendency of fault is not obvious enough, the pre-warning of system will be lagged behind or even cannot forecast an actual time step of malfunction.

4.3 Fault prediction based on particle filter and support vector machine integration framework

On the one hand, the calculation of weighted fault probability of Particle Filter for the fault-proneness prediction needs multi-step looping of computation with large numbers of particles and costs a lot of operating time. On the other hand, the utilization of residual error improves the Particle Filter based fault prediction to be much simpler but has strong prediction ability. Nevertheless, it cannot accurately predict the corresponding time step of malfunction relatively. Thus, we propose a new fault prediction method on the basis of Particle Filter and Support Vector Machine integration framework to combine strengths and avoid weaknesses. In this method, we firstly extract the system state parameters of particle filter in middle process from training particles set. We make use of process parameters as characteristic parameters, and then design training samples and corresponding output tags to train the model of Support Vector Machine. Fault prediction method of Particle Filter and Support Vector Machine (PF-SVM) integration framework is shown in Fig.7:

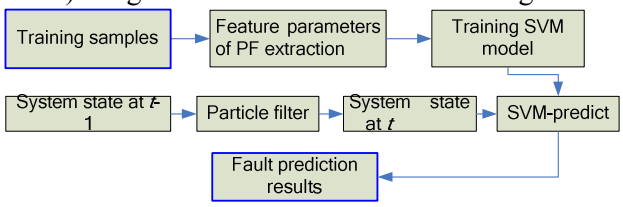


Fig.7 PF-SVM integrated framework

Instead of using the measured parameters to directly train the models, we take advantages of the Particle Filter to extract and estimate the process parameters for training the prediction model of Support Vector Machine.

The main advantages of this integration framework are list as following:

Firstly, since the characteristic parameters of its left wheel velocity, right wheel velocity, and both logical velocity are not obvious enough to indicate the tendency of fault, we should further extract some other characteristic parameters. Thus, we make use of particle filter to estimate parameters of system states and adopt the observation variables and state variables as features that can describe fault-proneness of the system because they are comparatively more obvious than the observed velocities.

Secondly, on the basis of the randomness of the "particles" of particle filter, we obtain different estimation parameters each time; thus we can gain the desired size of samples by multiple operating so as to enrich and improve our kinds and quantities of characteristic parameter of the training samples.

The Particle Filter and SVM integration framework based fault prediction is realized in mainly two steps:

1. Sample collecting and model training: to train SVM model in the first place, we use the training characteristic matrix in this method as the parameters of the system state estimation in Particle Filter. And the output of training tags are the prediction results of fault modes as shown in the left part of Fig.8.

2. Fault prediction: using the algorithm Particle Filter we directly forecast the parameters of system state by making use of the function ***svmpredict***(…) in Support Vector Machine toolbox as shown in the right part of Fig.8.

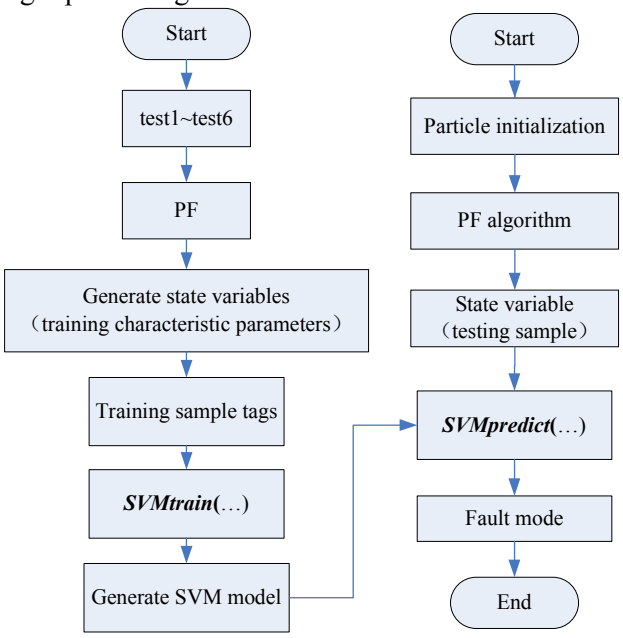


Fig.8 PF-SVM integrated flow chart

The fault prediction of time step using the method of Particle Filter and Support Vector Machine (PF-SVM) integration framework is relatively precise, comparatively simpler and practical with low time complexity. The problem of unobvious features with characteristic matrix is solved through enlarging the training sample because one of the great advantages of Support Vector Machine model is that its learning ability keeps working well no matter how large the sample size is. Furthermore, it is easy to enlarge the parameters sets of training samples because they can be produced and can be assured of the diversity by the randomness of particles.

5 Experiments analyses of fault prediction for robot dead reckoning system MORCS-1

For the convenience of expression, We label these four kinds of algorithms in this paper: algorithm 1: A particle filter based algorithm for fault prediction mentioned in 3.3-3.4; algorithm 2: A weighted fault probability based Support Vector Machine for fault prediction mentioned in 4.1; algorithm 3: Improved particle filter using residual error for fault prediction mentioned in 4.2; algorithm 4: A particle filter and Support Vector Machine integration framework for fault-proneness prediction mentioned in 4.3.

5.1 Sample data for SVM

To construct a complete library of training samples for the fault-proneness prediction of the robot dead reckoning system, in which the fault space has 2^3=8 kinds of fault modes (types), we have to set 8 kinds of sample tags. In algorithm 2, input characteristic parameter for the untrained Support Vector Machine is the weighted fault probability, while in algorithm 4 the input of the untrained model of Support Vector Machine is the process parameters of system states and the observation parameters of estimation by the Particle Filter. The output tags of both algorithm 2 and 4 are the fault modes, or the fault types. We record the logical inputs of both left and right wheel encoders and collect the actual output data of these wheel encoders and gyroscope, from which we will select parts of them as the training data of the model, and the rest as the testing data. In the collection of training samples, the entire types of fault modes should be included. And then the theory of Particle Filter is applied to extract weighted fault probability and other related parameters for different algorithms in these collections of training sample. To obtain the sample tags in Support Vector Machine training, we should bring ahead of time for k time steps to warn the actual fault mode corresponding to the training sample (such as k=5), that is, we should mark it out immediately to obtain the fault prediction sample tags when its relevant variation tendency of fault appears.

Considering the transformation relationships among fault types (shown in Fig.4) and their completeness in sample sets, we combine the actual operating of the robot dead reckoning system and conclude that when one of the sensors go wrong and cannot be repaired spontaneously, the system would transform into another fault mode bringing with its original malfunction. For this reason, we select the set of sample data of the fault transformation relationships, test1~test6, as shown in Fig.5, to cover the entire fault types. Possible fault types of the robot dead reckoning system can be one of them or their sub-cases, though there is redundancy exist among the sequences of fault modes in Fig.5. If the

actual fault type is only a part of one certain training sample, the Support Vector Machine model can also predict the corresponding results because the training samples almost include the entire transformation information among all the fault modes.

5.2 Experiments platform for robot dead reckoning system fault prediction

Software simulation allow to approximate the behavior of complex systems under several scenario conditions[19]. In this paper, we utilize Matlab as our experiment platform for simulation, and apply four different algorithms to sample its real-time data of sensors and predict the system state of fault with MORCS-1 robot. Our main interface of the program and its relevant sub-blocks are shown in Fig.9.

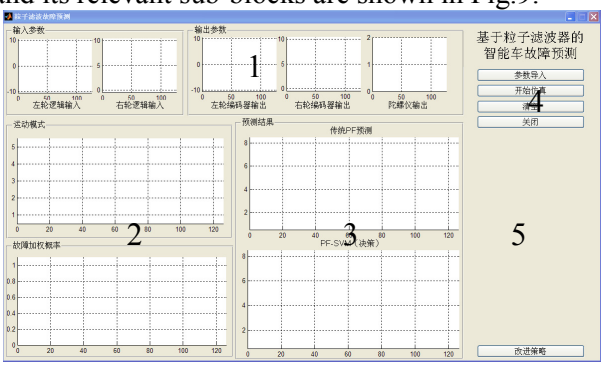
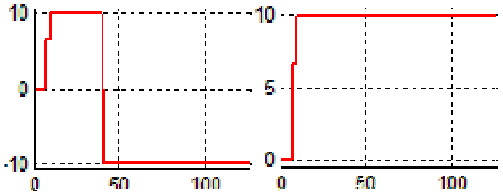


Fig.9 The main interface for simulation and analysis (1. Input and output blocks, 2. Prediction of process parameters block, 3. Prediction results block, 4. Control block, 5. Parameters description block)

In the whole process of the simulation experiment, block3 makes decisions for fault prediction by using the weighted fault probability and output of block2. Firstly we separately adopt the prediction scheme of algorithm1 and algorithm2, then click the button of "Improved Scheme" on the right side of the bottom to run the two improved schemes of algorithm3 and algorithm4 for fault prediction in block2; they don't have to calculate the weighted fault probability.

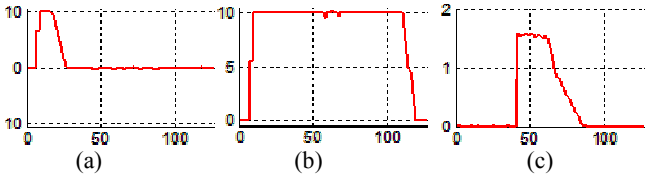
5.3 Experiments analysis for fault prediction method

In order to verify the fault prediction results of the 4 kinds of algorithms for robot dead reckoning system, we adjust MORCS-1 in different motion states to design and simulate fault modes. In Fig.10, there are the input parameters of both left and right wheel encoders. In Fig.11, there are the real-time output parameters of both wheel encoders and gyroscope of MORCS-1. We judge the motion mode of MORCS-1 by setting the velocities of both wheel encoders as shown in Fig.12, where its vertical coordinate 1 represents the linear motion, and vertical coordinate 2 represents the motion revolving about the right wheel while vertical coordinate 3 represents the motion rounding about the left wheel, and vertical coordinate 4 represents other motion mode, vertical coordinate 5 represents the state of rest, as described in 3.2. By calculating the parameters of input and output, we obtain Fig.12 where MORCS-1is in the state of rest at first, and then the input of both left and right wheels are equivalent linear motion, while in the rest states are kinds of curvilinear motions (mode 4).



(a) Left wheel (b) Right wheel

Fig.10 Logical input parameters



(a) (b) (c)

Fig.11 Output parameters

(a) Left wheel encoder (b) Right wheel encoder (c) Gyroscope

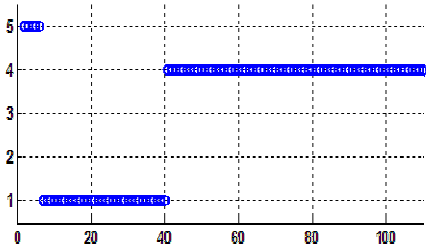


Fig.12 Motion modes

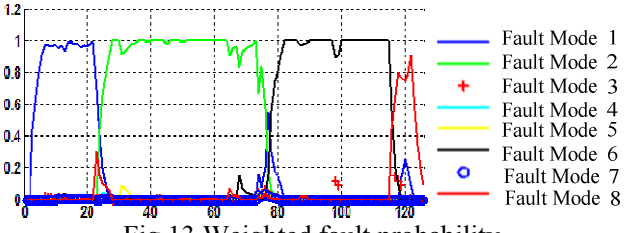
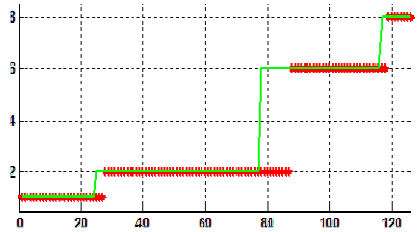


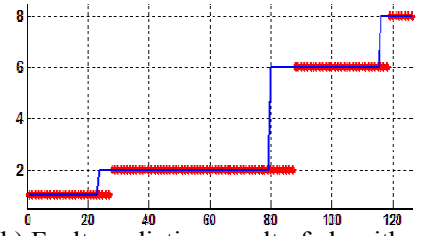
Fig.13 Weighted fault probability

As shown in Fig.13, to calculate the weighted fault probability by algorithm1, we set the threshold of weighted fault probability as 0.5 at first, in other word, when the weighted probability of a certain fault mode is greater than 0.5, this mode will be judged as the fault that would appear after several time steps. We can see that fault mode2 occurred at

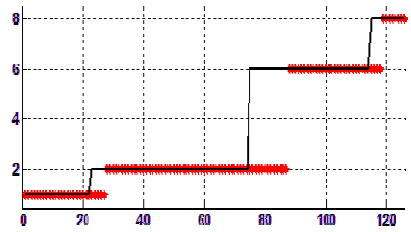
time step 25, fault mode5 appeared at time step 75, and fault mode8 happened at time step 118. Details of the entire fault modes for algorithm1 are listed in Fig.14 (a). It should be noted that the horizontal axis denotes time step, and vertical axis denotes fault mode. From the result, the pre-alarming are made at each turn although for the fault at time step 119, it warn only one time step in advance (giving an pre-warning at time step 118). The experiment data of algorithm2 for the fault-proneness prediction is shown in Fig.14 (b); and it was pre-alarming in time as well before each fault occurring, and for the fault at time step 119, it alarm less than 5 time steps in advance, but it was comparatively better than algorithm1. The fault prediction result of algorithm3 is shown in Fig.14(c), its pre-warning was made the same as above, but for the fault at time step 119, it warn nearly 5 time steps in advance, leaving enough time for emergency management; and at time step 89, its pre-alarm was more than 10 time steps ahead. This fact indicated that at time step 89, the variation tendency of fault is obvious, while at time step 119, it is relatively moderate. Last but not the least, with its fault prediction results of algorithm4 shown in Fig.14 (d), we can find that its pre-alarm was even better than the other algorithms: all the predictions are appropriately 5 time steps ahead than the real-time occurring of faults, that is to say, the stability of algorithm4 is the highest in predicting the time steps of fault occurring in experiments. Considering the fact that fault prediction of time steps ahead of schedule is directly related to the variation tendency of faults by computing its sampling data, we can conclude that these four fault prediction methods are all efficient, especially for algorithm4.



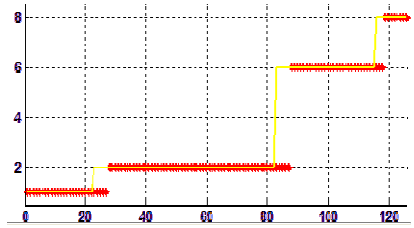
(a) Fault prediction result of algorithm 1



(b) Fault prediction result of algorithm 2



(c) Fault prediction result of algorithm 3



(d) Fault prediction result of algorithm 4

Fig.14 Fault prediction results of algorithms

In this simulation experiment of these algorithms, we have obtained better effectiveness and efficiency. Nevertheless, we define the "prediction accuracy" as the ratio of the correctly predicted samples and total testing samples, or it means the prediction data compared with the real-time data after all the operating ends. For this purpose, ten independent sampling experiments are carried out to test the four algorithms of their prediction accuracy. And we can find that the average accuracy is above 95% as shown in Fig.15 where they are close to each other, but only algorithm4 has a tendency of ascendancy.

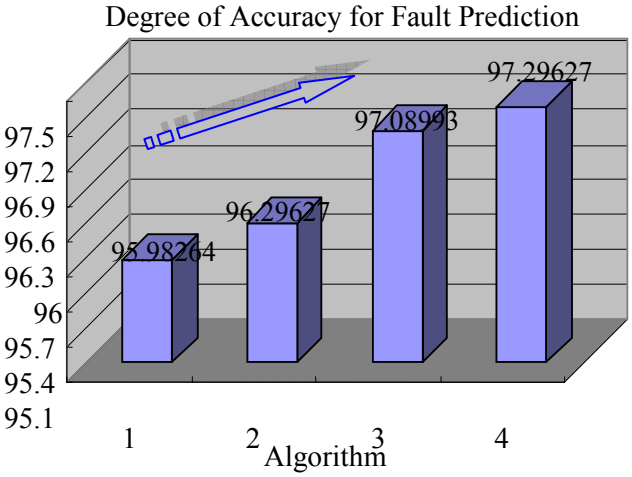


Fig.15 Comparisons of these four algorithms on the degree of fault prediction accuracy

Table.5 Run time comparison (Time unit: s)

Algorithm		1	2	3	4	5
1	ALG 1	11.499	10.744	10.225	11.704	10.189
	ALG 2					
	ALG 3	6.006	6.041	6.141	8.071	6.224
	ALG 4	6.402	6.713	6.502	8.538	6.673
2	ALG 1	10.583	11.044	11.086	10.767	10.747
	ALG 2					

3	ALG 3	6.282	6.362	6.449	6.167	6.121	
	ALG 4	6.715	6.873	7.048	6.761	6.562	
	ALG 1 ALG 2	10.504	10.604	10.846	11.678	10.976	
	ALG 3	6.171	6.148	6.022	6.309	6.351	
	ALG 4	6.395	6.796	6.392	7.03	6.805	
Algorithm		6	7	8	9	10	Average
1	ALG 1 ALG 2	9.949	10.197	11.026	10.275	10.586	10.639
	ALG 3	6.323	6.66	6.37	6.077	6.139	6.405
	ALG 4	6.942	6.624	7.372	6.436	6.659	6.886
2	ALG 1 ALG 2	10.389	10.195	10.625	10.271	10.305	10.601
	ALG 3	6.362	6.077	6.013	6.621	6.192	6.264
	ALG 4	6.943	6.282	6.5	7.083	6.53	6.729
3	ALG 1 ALG 2	9.939	10.953	10.737	9.927	10.294	10.645
	ALG 3	5.918	6.131	6.154	5.789	6.250	6.124
	ALG 4	6.313	6.519	6.567	6.029	6.789	6.563

To statistically compare the runtime of these four algorithms, we conduct three independent groups of experiments, for each group we carried out ten independent experiments and had their records each runtime. From the obtained experiment data in table 2, we can see that the runtime of algorithm 1 and 2 are much similar as our expectation because their prediction parameters are much the same while only their decision-making methods are different. So we consider algorithm 1 and 2 as the same type. What's more, the computing time of algorithm 1 and 2 are comparatively complex as shown in table 5 while algorithm3 cost much less runtime. Although the time-consuming of algorithm4 is slightly more than algorithm3, it is generally acceptable for most of the systems. Thus, we can conclude that algorithm4 is the best of these four algorithms not only for its higher average accuracy of prediction but also for the stability of the pre-warning time steps for fault prediction with a relatively better efficiency.

The proposed algorithm is applied in MORCS-1 successfully, and the mobile robot can predict fault in real time, so the satisfactory application results are obtained.

6 Conclusion and prospective

Algorithm 1 is based on the traditional particle filter prediction framework whose extracted characteristic parameter is the weighted fault probability, and its output accuracy is basically stable for nearly all the testing data; algorithm 2 has a relatively higher fault prediction accuracy than algorithm1, indicating that the construction of SVM model to replace the fixed setting of threshold of algorithm 2 is efficient. However, both of these two algorithms cost too much runtime for the extracting of weight fault probability. Algorithm 3 uses residual error to simplify the computation, thus improves algorithm runtime. Algorithm 3 is not only simple and easy to realize but also fast to operate with lower false rate of alarming. But compared with algorithm 4, time step for fault prediction of algorithm 3 is relatively imprecise because it mainly depends on variation tendency of data. Algorithm 4 improves algorithm 1, 2 and 3 respectively by integrating all of their advantages. Algorithm 4 can calculate the time step of fault prediction with the highest accuracy and speed. Furthermore, if a little bit weaker was shown in stability of algorithm 4, we could solve the problem by improving the sets of training samples or even retrain for the Support Vector Machine.

At last, we list some needed improvements of these experiments as following:

a) Firstly, in this article, the experiments supposed that the robot did not moved on complexity terrain for the purpose of simplifying the model;

b) Secondly, when applying the basic algorithm of particle filter to fault-proneness prediction, the problems of particle degeneracy and its efficiency should be taken into consideration.

c) Thirdly, the fault-proneness prediction often lacks a universal type. In this article, we utilize the SVM to improve particle filter of fault-proneness prediction for the purpose of generalization. In the future, other techniques of machine learning will be taken into consideration to complete a generalized fault prediction framework.

In brief, research of the fault-proneness prediction for hybrid systems is still in full swing and problems mentioned above need to be solved a step further.

Acknowledge

The authors would like to acknowledge the postdoctoral granted financial support from China postdoctoral science foundation (20110491272), National Natural Science Foundation (61104014) and Education Department of Hunan research project (11B070).

References:

[1] Zhijie Zhou, Changhua Hu, Hongdong Fan, et al. Fault Prediction of the Nonlinear System with Uncertainty, *Simulation Modeling Practice and Theory*, Vol.16, 2008, pp.690-703.

[2] Hu Chang Hua, Zhang Qi, Qiao Yu Kun. Strong Tracking Particle Filter with Application to Fault Prediction. *Acta Automatica Sinica*, Vol.34, No.12, 2008, pp.1522-1528.

[3] Catal Cagatay, Software Fault Prediction: A Literature Review and Current Trends. *Expert Systems with Applications,* Vol.38, No.4, 2011, pp.4626-4636.

[4] Hamdi-Cherif, A.. Intelligent control and biological regulation for bioinformatics, *International Journal of Mathematical Models and Methods In Applied Sciences*, Vol. 4, No.2, 2010, 93-104.

[5] Di Martino Sergio, Ferrucci Filomena, Gravino Carmine, Sarro Federica. A Genetic Algorithm to Configure Support Vector Machines for Predicting Fault-Prone Components. *Lecture Notes in Computer Science*. Vol.6759, 2011, pp 247-261.

[6] Zhang Qi, Hu Chang Hua, Qiao Yu Kun, Cai Yan Ning. Fault Prediction Algorithm Based on Stochastic Perturbation Particle Filter, *Control and Decision*, 2009, Vol.24, No.2, pp284-288.

[7] Chen Chaochao, Zhang, Bin, Vachtsevanos George, Orchard, Marcos. Machine Condition Prediction Based on Adaptive Neuro-Fuzzy and High-order Particle Filtering. *IEEE Transactions on Industrial Electronics,* Vol.58, No.9, 2011, pp4353-4364.

[8] Li Baoan, Liu Zhihua, Li Xinjun. Research of UAV Engine Fault Prediction Based on Particle Filter. *Proceedings of the 9th International Conference on Electronic Measurement & Instruments (ICEMI 2009)*, 2009, pp.4/813-17.

[9] Zhengguo Xu, Yindong Ji, Donghua Zhou. A new real-time reliability prediction method for dynamic systems based on on-line fault prediction, *IEEE Transactions on Reliability,* Vol.58, No.3,2009,pp.523-538.

[10] Iker Gondra, Applying Machine Learning to Software Fault-Proneness Prediction, *The Journal of Systems and Software*, Vol.81, 2008, pp.186-195.

[11] Ni J., Zhang C., Yang S. X., An Adaptive Approach Based on KPCA and SVM for Real-Time Fault Diagnosis of HVCBs, *IEEE Transactions on Power Delivery*, Vol.26 , No.3, 2011, pp.1960-1971.

[12] Yan Zhang, Bide Zhang, Yuchun Yuan, Zichun Pei, Transformer Fault Prediction Based on Support Vector Machine, *International Conference on Computer Engineering and Technology (ICCET),* 2010, Vol. 3, 2010, pp. 513-516.

[13] Qin Li-Na, Software Reliability Prediction Model Based on PSO and SVM. *International Conference on Consumer Electronics, Communications and Networks (CECNet),* 2011, pp.5236-5239.

[14] Adrian Smith, Sequential Monte Carlo Methods in Practice, *New York: Springer-Verlag,* 2001.

[15] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking, *IEEE Transactions on Signal Processing,* vol.50, No. 2, 2002. pp.174-188.

[16] Byoung Wook Choi, Dong Gwan Shin, Jeong Ho Park, Soo Yeong Yi, Seet Gerald. Real-time control architecture using Xenomai for intelligent service robot in USN environment, *Journal of Intelligent Service Robotics*, Vol. 2, No.2, 2009, pp.139-151.

[17] Zhang Lei, Li Xingshan, Yu Jinsong, Liao Canxing. A Fault Prognostic Algorithm Based on Hybrid System Particle Filter and Dual Estimation, *Acta Aeronautica Et Astronautica Sinica,* Vol.30, No.7, 2009,pp.1277-1283.

[18] Chih Chung Chang, Chih Jen Lin, LIBSVM: a Library for Support Vector Machines, 2001. Software Available at: http://www.csie.ntu.edu. tw/~cjlin/libsvm.

[19] Neri F.. Software agents as a versatile simulation tool to model complex systems. *WSEAS Transactions on Information Science and Applications*, Vol. 7, 2010, pp. 609-618.