Fault Detection and Identification in a Mobile Robot Using Multiple Model Estimation and Neural Network

Puneet Goel, Göksel Dedeoglu, Stergios I. Roumeliotis, Gaurav S. Sukhatme*

puneet|dedeoglu|stergios|gaurav@robotics.usc.edu

Robotics Research Laboratory
Institute for Robotics and Intelligent Systems
Department of Computer Science
University of Southern California
Los Angeles, CA 90089-0781

Abstract

We propose a method to detect and identify faults in wheeled mobile robots. The idea behind the method is to use adaptive estimation to predict the outcome of several faults, and to learn them collectively as a failure pattern. Models of the system behavior under each type of fault are embedded in multiple parallel Kalman Filter (KF) estimators. Each KF is tuned to a particular fault and predicts, using its embedded model, the expected values for the sensor readings. The residual, the difference between the predicted readings (based on certain assumptions for the system model and the sensor models) and the actual sensor readings, is used as an indicator of how well each filter is performing. A backpropagation Neural Network processes this set of residuals as a pattern and decides which fault has occurred, that is, which filter is better tuned to the correct state of the mobile robot. The technique has been implemented on a physical robot and results from experiments are discussed.

1 Introduction

Fault detection and identification (FDI) are important problems in the development of reliable, robust mobile robots. In this paper we present an execution of the technique called Multiple Model Adaptive Estimation (MMAE) to the case of the FDI problem in mobile robots, complemented by a neural network based pattern recognition approach. The aim of this study is to show that it is possible to detect and identify both sensor and mechanical failures on a mobile robot platform by means of analyzing the collective output of a bank of Kalman Filters.

We have used a bank of Kalman Filters to model various faults. Kalman filtering [7] is a well known technique for state and parameter estimation. It is a recursive estimation procedure that uses sequential measurement data sets. Prior knowledge of the state (expressed by the covariance matrix) is improved at each step as prior state estimates are used for prediction and new measurement for subsequent state update. An artificial Neural Network, trained with the well known error backpropagation algorithm[3], is used for processing the set of filter residuals given by the filters as a pattern and deciding which fault has occurred, that is, which filter we should believe in.

2 Problem Definition and Algorithm

2.1 Task Definition

Fault tolerant behavior refers to automatic detection and identification of faults as well as the ability to continue functioning after a fault has occurred. Our aim is to *detect* (on time) when the fault occurs and *identify* it among a set of possible failures when the model of system is available. Detection of a fault is relatively simple and can be done using only one filter, the one representing the nominal model of the system. If the values estimated by the filter deviate largely from the measurements, something has gone wrong. The critical part is to *identify* what has gone wrong.

To this end, we used a *bank* of Kalman filters. Each filter assumes that a different type of failure has occurred and uses the appropriate system and sensor

*contact author for correspondence

Figure 1: The Pioneer AT and simulation of Flat Tires

model to predict the behavior of the robot based on certain assumptions for possible failures. Specifically, we apply the technique to three independent sensor failures and four mechanical failures. Sensor failures include 'hard' failures of the gyroscope, left encoders and right encoders on the robot (see figure 1). Mechanical failures include one flat tire on the left, two flat tires on the left, one flat tire on the right and two flat tires on the right. The non-failure case filter predicts the normal behavior. So in total we have eight filters.

The problem at hand is how to decide which filter to choose. The naive approach would be to simply check the corresponding residuals for each filter and believe the one with the minimum residual. Yet, real data is always noisy and this naive approach is not of much help since it does not contain any information about the validity of the residual information. The residual (innovation) covariance is required in order to be able to compare two residual vectors originating from two different filters. Figure 2 shows a typical set of residuals from the bank of Kalman Filters (Different grey-scales are used to distinguish between the residuals from multiple Kalman Filters). As expected the residuals are quite noisy. It is almost impossible to even visually distinguish the residuals. The ones from the tuned filter should obey the zero-mean white assumption. Possible improvements could be achieved by using some kind of Bayesian Probability methods or artificial neural network. Maybeck [9] has shown that one way to achieve fast response to failures is beta elimination while using Bayesian Probability. Here we use a backpropagation neural network [3] to identify failures.

2.2 Algorithm Definition

The Pioneer AT used for experiments is a four wheeled robot shown in Figure 1. The wheels on the same side are mechanically coupled. The encoders return only two distinct speeds; one for the right pair of wheels and other for the left pair of wheels. The

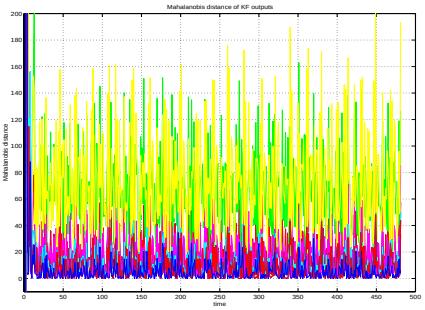

Figure 2: Residuals from the bank of KFs. Different grey-scales are used to distinguish between the residuals from multiple Kalman Filters.

kinematics of the Pioneer AT are given in Equations (1)-(2).

$$\theta_{k+1} = \theta_k + \dot{\theta}\ dt \tag{1}$$

$$\dot{\theta} = \frac{v_R - v_L}{l} \qquad v_{tot} = \frac{v_R + v_L}{2} \tag{2}$$

where l is the vehicle axle length, v_L and v_R are the velocities of the left and right wheels respectively. $\dot{\theta}$ is the yaw rate of the robot in the x-y plane and θ is the angle between the vehicle axle and x axis. For the experiments reported here, the frame of reference is chosen in such a way that the start location of the robot is the origin facing in the positive y direction. There are three sensors on-board the Pioneer AT robot, two encoders which return left and right wheel velocities and a gyroscope that gives the rate of change of the angle in x-y plane. The kinematic quantities of interest are shown in Figure 3.

If we expand the first part of equation 2 we get

$$\dot{\theta} = \frac{\omega_R\ R_r - \omega_L\ R_l}{l} = \frac{\omega_L\ \hat{R}}{l\ \frac{\hat{R}}{R_r}} - \frac{\omega_R\ \hat{R}}{l\ \frac{\hat{R}}{R_l}} \tag{3}$$

where ω_R and ω_L are the angular velocities of the right and left wheels respectively. R_r and R_r are the radii

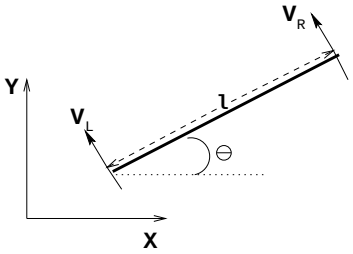

Figure 3: The Robot Kinematics

of the left and right wheels respectively. The velocities returned by the encoders are equivalent to $\omega_L \hat{R}$ and $\omega_R \hat{R}$. When a tire goes flat we need to estimate R_l and R_r. This estimation is done off-line by making all the tires flat and then taking the ratio of distance traveled in this situation to the distance traveled if no tire is flat. This gives an estimate of the radius of a flat tire. The flat tire is simulated by first making the tires thick by wrapping paper towels around them, and then removing them. The tire after removal of the towels is assumed to be the flat tire (Figure 1).

Sensor failures are simulated by assuming that a sensor gets stuck at some particular value (hard failure) and keeps returning that value once the failure has occurred. A model for each such case is embedded within a Kalman filter. When the measurements are received they are fed into these different filters and each filter outputs residuals based on the model encapsulated.

In the experiments reported here, the measurement vector is composed of the two translational velocities of the left and right wheels and the yaw rate of the chassis as measured by the gyroscope. $\hat{\mathbf{x}}$ is the state estimate vector, $\mathbf{z}$ is the measurement vector, $\mathbf{r}$ is the residual vector and $\hat{\mathbf{z}}$ is the measurement estimate.

$$\mathbf{z} = [v_L \; v_R \; \dot{\theta}]^T \quad \hat{\mathbf{z}} = \hat{\mathbf{x}} = [\hat{v}_L \; \hat{v}_R \; \hat{\dot{\theta}}]^T \quad \mathbf{r} = \mathbf{z} - \hat{\mathbf{z}} \quad (4)$$

Kalman filtering is a repetitive process consisting of two consecutive steps **propagation** and **update**. The equations for the propagation step are:

$$\hat{\mathbf{x}}_{k+1/k} = \mathbf{\Phi} \; \hat{\mathbf{x}}_{k/k} \quad (5)$$

$$\mathbf{P}_{k+1/k} = \mathbf{\Phi} \; \mathbf{P}_{k/k} \; \mathbf{\Phi}^T \; + \; \mathbf{Q} \quad (6)$$

The system matrix $\mathbf{\Phi}$ is given by:

$$\mathbf{\Phi} = \begin{bmatrix} \alpha & b & c \\ d & \beta & e \\ -1/l & 1/l & 0 \end{bmatrix}$$

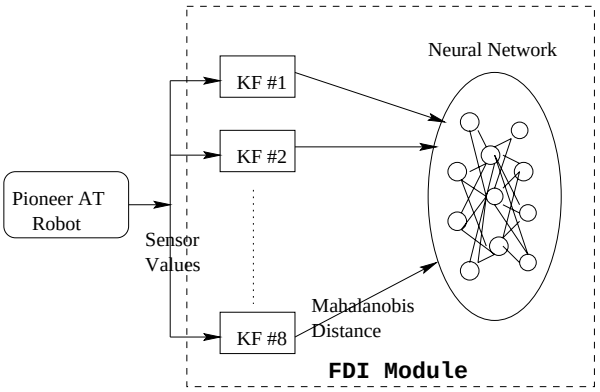

Figure 4: Pictorial Representation of the Algorithm

The equations for update are

$$\mathbf{S} = \mathbf{H} \; \mathbf{P} \; \mathbf{H}^T + \mathbf{R} \quad (7)$$

$$\mathbf{K} = \mathbf{P}_{k+1/k} \; \mathbf{H}^T (\mathbf{S})^{-1} \quad (8)$$

$$\hat{\mathbf{x}}_{k+1/k+1} = \hat{\mathbf{x}}_{k+1/k} + \mathbf{K} \; \mathbf{r} \quad (9)$$

$$\mathbf{P}_{k+1/k+1} = (\mathbf{I} - \mathbf{K} \; \mathbf{H}) \; \mathbf{P}_{k+1/k} \quad (10)$$

where $\mathbf{S}$ is the residual covariance matrix, $\mathbf{P}$ is the state estimate covariance matrix, $\mathbf{Q}$ is the system noise covariance matrix, $\mathbf{K}$ is the Kalman gain matrix and $\mathbf{R}$ is the measurement noise covariance matrix. $\mathbf{H}$ is used to convert the estimated state vector into the format in which the measurements are obtained. α and β are unity and b, c, d and e are zero for the filter with nominal behavior and the filters modeling sensor failures while they take different values for the filters embedded with mechanical failure model. For example, if the left encoder fails the estimation of left wheel velocity will be done using values returned by right encoder and gyroscope and we will get $\alpha = 0$, $b = 1$ and $c =$ -l.

The system noise covariance matrix $\mathbf{Q}$ is determined empirically. Different sets of experimental data were processed to calculate the system driving noise. The values of measurement noise matrix $\mathbf{R}$ are based on sensor specifications as well as empirical observations.

Each filter modeling a sensor failure has a different $\mathbf{R}$ matrix since this is the matrix that represents noise in sensor readings. If an encoder fails the value corresponding to it in the $\mathbf{R}$ matrix rises. The matrix $\mathbf{H}$ is a 3x3 identity matrix.

Figure 4 shows a pictorial representation of the algorithm. The *Mahalanobis distance* that forms the

input to the neural network is given by:

$$\mathbf{Dis} = \mathbf{r}^T \ \mathbf{S}^{-1}\mathbf{r} \tag{11}$$

where $\mathbf{r}$ represents the residual vector for $\mathbf{v}_L$ $\mathbf{v}_R$ and $\dot{\theta}$ respectively while $\mathbf{S}$ is the covariance matrix associated with the residual vector.

The algorithm involves following steps:

1. Measurements from encoders and gyroscope are fed to the bank of filters.

2. Each filter produces a residual and covariance matrix associated with it.

3. Calculate the Mahalanobis distance for each filter using 11.

4. For each filter add this distance to the sum of distances from previous iterations for that filter. Store the new sum.

5. Calculate the average summed value for each filter.

6. Normalize the summed value of each filter by dividing it by the maximum sum. This forms the input to the Neural Net.

7. The neural network processes the pattern and produces the confidence level for each filter.

8. The filter with the highest confidence level is chosen as the best match.

The neural network is trained off-line using the converged values of the mean of the Mahalanobis distance aggregated over all the previous values. Nine patterns were used to train the network using a standard backpropagation formulation:

$$(\triangle \mathbf{w}_{ji})_n = \eta\delta_j \mathbf{o}_i \ + \ \gamma(\triangle \mathbf{w}_{ji})_{n-1} \tag{12}$$

$$\delta_j = \mathbf{o}_j(\mathbf{1} - \mathbf{o}_j)(\mathbf{t}_j - \mathbf{o}_j) \quad for\ output\ unit \tag{13}$$

$$\delta_j = \mathbf{o}_j(\mathbf{1} - \mathbf{o}_j)\Sigma_k \delta_k \mathbf{w}_{kj} \quad otherwise \tag{14}$$

where $\triangle \mathbf{w}_{ji}$ represents the change in the weight of the link joining j^{th} and i^{th} units, δ represents the change in weights of output and hidden units, and η and γ denote learning rate and momentum respectively. o_j represents the output of the j^{th} unit while t_j is its target output.

3 Experimental Evaluation

3.1 Methodology

The aim of this study is to show that it is possible to detect and identify both sensor and mechanical failures on a mobile robot platform by means of analyzing the collective output of a bank of Kalman filters. To this end, we performed a number of experiments on a real robot, which was programmed to follow straight (at 300 mm/sec translational, and 0 degrees/sec rotational velocities) and circular (300 mm/sec translational, and 30 degrees/sec rotational velocities) trajectories. All sensor data (left & right wheel velocities measured by the encoders and the gyroscope's measurement of the rotational velocity) was collected, to be analyzed later by simulating the Kalman Filters off-line.

In order to define a repeatable and comparable pattern over all Kalman filters' outputs, we scaled all outputs to fall into the interval [0, 1] by simply dividing them by the largest filter value computed at that moment. Our training patterns were taken from the converged values of these outputs, summarized in a table in figure 5.

For the gyro failure case, we have had an interesting observation. Nominal case cannot be distinguished from the gyro failure case (if the robot is moving straight), which can be intuitively explained as follows: If the robot is running along a straight line, one would expect the gyro to return a mean value of zero rotational velocity, corrupted by some Gaussian noise. Yet, if the gyro fails, it will return zero readings no matter what the robots movement is (hard sensor failure). The assumption is that hard failure is loss of signal corresponding to a fixed value measurement (which we have assumed to be zero here). For this reason, it is not possible to identify gyro failures by executing straight line trajectories. In order to capture this failure mode, we experimented with two extra cases (hence, patterns), corresponding to clockwise and counter-clockwise turns, as depicted in the table of Figure 5. That is why we had nine training patterns in all, two for gyro failure and one each for the other cases.

A multilayer feed-forward network with a single hidden layer was used to learn the patterns described above. At each layer, the number of neurons taken was eight, which is the number of cases we were trying to identify (1: Nominal case, 2: Gyro failure, 3-4: Left-Right encoder failures, 5-6: One-two flat tires on the left, 7-8: One-two flat tires on the right) The training algorithm implemented the error backpropagation rule

Correct Filter	Nominal Filter	1 Left Flat Tire	2 Left Flat Tires	1 Right Flat Tire	2 Right Flat Tires	Gyro Fails	Left Enc. Fails	Right Enc Fails
Nominal Filter	0.135	0.3	0.98	0.3	1.0	0.2	0.12	0.09
1 Left Flat Tire	0.25	0.4	1.0	0.39	0.9	0.38	0.21	0.16
2 Left Flat Tires	0.3	0.45	1.0	0.41	0.85	0.45	0.22	0.23
1 Right Flat Tire	0.37	0.49	0.91	0.5	1.0	0.55	0.32	0.23
2 Right Flat Tires	0.32	0.43	0.8	0.46	1.0	0.47	0.22	0.26
Gyro Fails ~Clock	0.23	0.23	0.24	0.38	1.0	0.34	0.04	0.3
Gyro Fails Clock	0.1	0.28	1.0	0.11	0.13	0.15	0.06	0.1
Left Enc. Fails	0.03	0.08	0.24	0.23	1.0	0.04	0.04	0.0
Right Enc Fails	0.02	0.22	1.0	0.07	0.23	0.03	0.0	0.03

Figure 5: The converged mean values of Mahalanobis distances which were used for training.

to adjust the weights of the fully connected neurons, defined as sigmoid threshold units. The learning rate was chosen as 0.5, and a momentum of 0.75 was added to the weight-update rule. An output layer neuron was considered to fire correctly for activation levels higher than 0.9. In all experiments, 700 epochs of training were found to be adequate for the proper learning of all ten patterns.

3.2 Results

In order to assess the performance of the trained network in recognizing failures, we have collected test data and observed its output. Since the Kalman filters update their outputs at 10 Hz, a typical test run of 20 seconds would generate 200 test patterns. In all we made sixteen such test runs. As we see in the top left graph of Figure 6, the output neuron for nominal case (no failures) fires at a distinctly higher rate starting from t=5 seconds, well before all the Kalman filters had settled into their final estimates. As a further test of the network's capability to handle noise, we have run the same experiment at a different speed of the mobile robot, for which no training patterns were provided. As seen in top right graph of the same figure, the network was still able to confidently identify the correct case, this time with a slight extra delay of 5 seconds, mainly due to the transition period which dominates filter outputs at the start.

The middle two graphs depict the network output for the two right tires flat case, at two different speeds. Unfortunately, the network is unable to differentiate between the one-flat tire and two-flat tires on the same side. We have observed the same effect for the left side too, leading us to the conclusion that we can not separately identify those cases with our current pattern generation and recognition setup. We can identify the side which has a flat tire but we cannot say how many (that is, one or two) tires are flat. Except for these two cases we never obtained any misclassification. The response time was also very good. In most cases it took less than 3 seconds to detect the fault.

The last row of Figure 6 shows network performance for left encoder failures, as experimented at different velocities. It can be seen that in both cases the network reliably identifies the failure within 3 seconds.

3.3 Discussion

First we point out the reason why a neural network (or some recognition algorithm) is required in order to choose the correct filter based on the residuals. The naive approach is to select the filter with minimal residual. Measurement noise is an argument against this approach. More importantly, as shown in the table provided in Figure 5, the filter with the least residual is not always the correct one. For example, in the first row, although the filter that is tuned to the current state of the vehicle is the nominal filter the minimum converged residual value comes from the filter that assumes a right encoder failure. This clearly shows the need for a technique more sophisticated than choosing the filter with least residual. We found a neural network to be capable of performing this task.

We have shown that by using a relatively simple

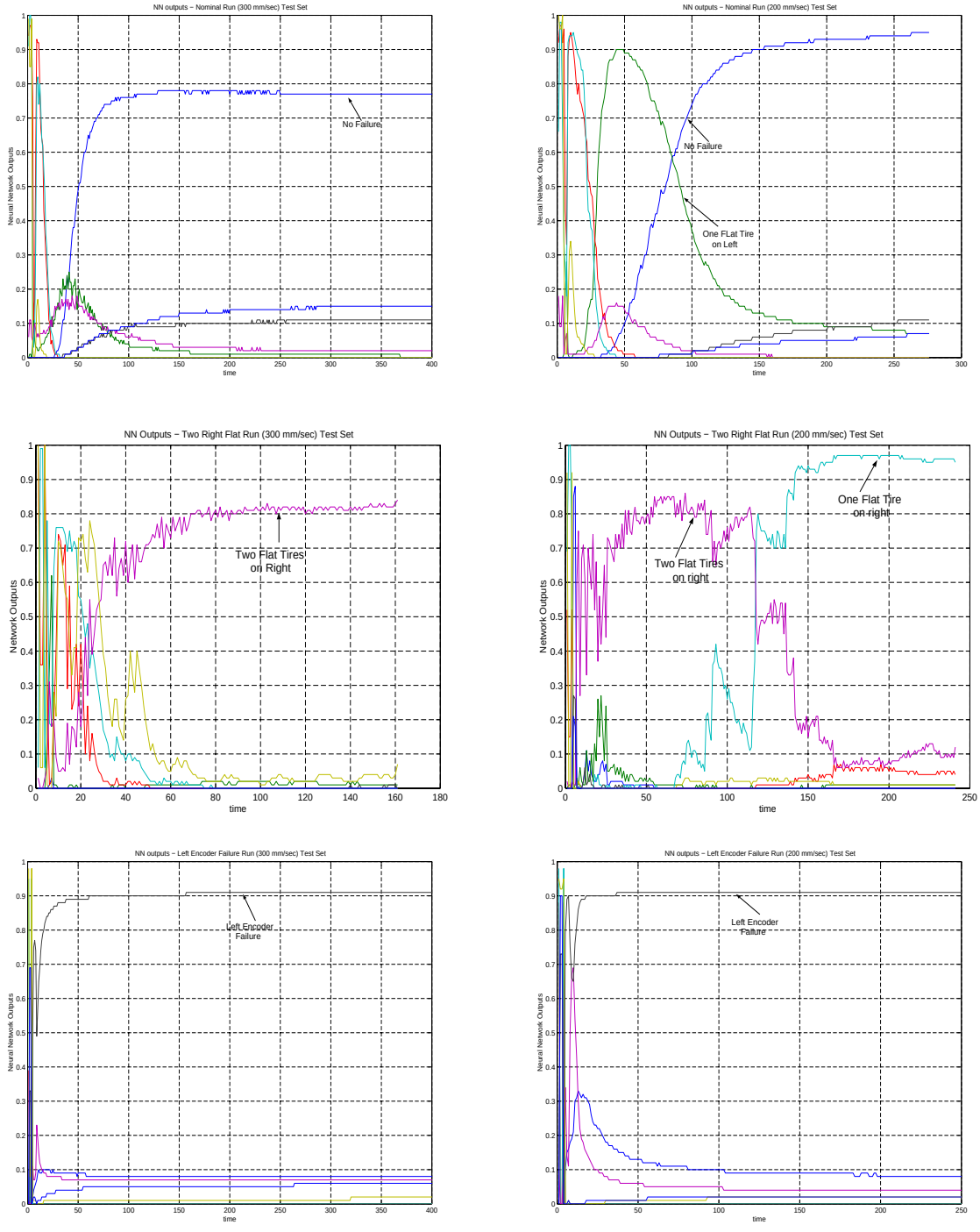

Figure 6: Neural Network Performance in various failure modes.

pattern recognition approach, both mechanical and sensor failures can be reliably detected and identified. As already stated, the tires on the same side of the mobile robot we used are tightly coupled. This omitted two cases of flat tires, as we can not distinguish between a front and rear flat tire on the same side. Therefore, we are left with only four possible flat tire combinations, namely one/two flat tires on the left/right. In addition to these mechanical failures, three sensors lead to a total number of seven failure cases. Having one normal and seven failure modes, we built a bank of eight Kalman Filters, each of which had an appropriate model embedded in it.

Although our initial intention was to identify four separate cases with regard to flat tires (i.e., one/two flat tires on the left/right), our experimental findings have shown the difficulty of this level of distinction. Instead, we were only able to identify the side on which the flat tire was present. We believe that this result is due to the extreme similarity between the residual signatures of cases involving one and two flat tires on the same side.

It should be noted that our training patterns were recorded at a specific velocity. However, this did not preclude the network from successfully identifying failures at different vehicle velocities. We believe that enriching the training set with additional signatures taken at various velocities would definitely improve the accuracy, and enable the network to differentiate patterns at a finer granularity. That is, the network shall be able to distinguish even between very closely related patterns.

We will also like to point out here that we decided to keep the number of Kalman filters and faults the same. It might be possible to identify larger number of faults using the same number of filters as the patterns for the new faults might differ from the existing patterns and hence the neural network can reliably learn them. The objective of this study however is not just detection and identification of faults but also to use the information about faults to reliably operate the robot even when a fault has occurred. In the future we plan to implement a feedback loop so that we can use the output from the correct Kalman Filter to reliably update the current information about the robot. Since we cannot currently embed more than one model in a filter, there is a one-to-one correspondence between faults and filters.

Currently the neural network we use has eight input units, eight hidden units and eight output units. We tried varying the number of hidden units. As we reduced the number of hidden units the performance deteriorated. It was observed that reducing the number of hidden units to less than five, leads to loss of generalizability, that is, if test patterns are collected for different vehicle speeds than the training patterns, the performance deteriorates considerably. Also there was a significant fluctuation in the performance for the test patterns collected at the same speed. It was also observed that the performance was acceptable with the number of hidden units greater than or equal to six.

In order to determine the behavior of the FDI module in case of unknown failures we experimented with a fault which was not in our training set. We modeled a fault where all four tires go flat during motion in a straight line. We fed the Mahalanobis distance of the residuals from eight Kalman filters to the network. Since all the four tires were flat we expected the same kind of patterns as would be given by the robot with no flat tire but moving at slower translational velocity (Translational Velocity = Rotational Velocity * Radius of Wheel). This is exactly what happened. The network classified the test data as nominal.

Future research is aimed at a better understanding of transitional failure modes as well as extensions to other sensor failure applications. Our ongoing research involves utilization of fault tolerant techniques in mapping and localization. Recovery from failure by altering the control strategy is also the subject of future work. The easiest (and least autonomous) solution is to stop, flag a fault and await human help. Other strategies include making guarded motions and switching to backup system.

4 Related Work

Using a bank of Kalman filters was pioneered by Magill [8] who used a parallel structure of estimators in order to estimate a sampled stochastic process. Subsequently Athans et al. [1] used a bank of Kalman filters that provided state estimates to an equal number of LQG compensators to provide control over different operating regimes of an aircraft. Each estimator relied on a set of system equations linearized about a different operating point. Later Maybeck et al. [10] used Further, in work by Maybeck et al. [11] [9] the multiple model adaptive estimation (MMAE) technique was used to reliably detect and identify sensor and actuator failures for aircraft.

In recent years Kalman filter based localization has become common practice [2] [14] [6] in the robotics literature. Since the MMAE technique relies upon a bank of Kalman filters it seems natural to apply it

to fault detection and identification in mobile robot systems. The important aspect of the method is to use analytical redundancy in the form of several system models (as opposed to say hardware redundancy which replicates hardware to identify a failure). A Kalman filter based framework provides a measure of the disparity (typically called a residual) between the measured sensor values and the values predicted by the model embedded within the filter. The residual is used in the filter to update the estimate and is an excellent indicator of failure.

A similar approach to ours has been used by [4] [5], in the context of space shuttle engine diagnostics and automated vehicle guidance systems. Previous work using MMAE applied to the case of mechanical failures (such as flat tires) on board mobile robots and independent sensor failures is due to Roumeliotis et al[12] [13]. The work done lacks intermingling of sensor failures and mechanical failures. Separate bank of filters were used for sensor failures and mechanical failures. Here, we are using only one bank of filters. Also the work differs in the technique used to decide for the tuned filter. Roumeliotis et al used probabilistic methods while we are using a neural network. The crux of our work lies in using same bank of filters for both kind of failures and then trying to decide which fault has actually occurred. It is a difficult task as a sensor failure and a mechanical failure can return the residual signatures that are very close to each other.

5 Conclusion

In this paper we have presented a Multiple Model Adaptive Estimation (MMAE) based technique for fault detection and identification using a neural network on-board a mobile robot. Experimental evidence is presented to show that the suggested technique works well for several different failures. The implementation described here is able to use measurements from several sensors. Detection and identification of faults is done by analyzing the signature of the residual produced by each filter.

One of the most important contributions of this work is the number and variety of faults considered. We have considered both mechanical and sensor failures at the same time, which is a difficult problem since signatures of residuals for these two different kinds of faults are very similar. It is shown that a good learning module, can distinguish closely related faults. Without using sophisticated techniques to analyze the residuals we are able to do detection of eight faults and identification of six faults. It should be noted that the method presented here is generalizable to more faults and increasingly sophisticated filters and residual processing. The method is applied to a Pioneer AT, a four wheeled robot, and can be easily extended to other mobile systems since it relies on simple kinematic descriptions.

Acknowledgments

This research is sponsored in part by contract #959816 from NASA/JPL and contract #F04701-97-C-0021 and #DAAE07-98-C-L028 from DARPA.

References

[1] M. Athans, D. Castanon, K.P. Dunn, C.S. Greene, W.H. Lee, N.R. Sandell Jr., and A.S. Whilsky. The stochastic control of the f-8c aircraft using a multiple model adaptive control (mmac) method-part i: Equilibrium flight. *IEEE Transactions on Automatic Control*, AC-22(5):768–780, October 1977.

[2] B. Barshan and H. F. Durrant-Whyte. Inertial navigation systems for mobile robots. *IEEE Transactions on Robotics and Automation*, 11(3):328–342, June 1995.

[3] C.M. Bishop. *Neural Networks for Pattern Recognintion*. Oxford University Press, 1995.

[4] R.K. Douglas, D.P. Malladi, R.H. Chen, D.L. Mingori, and J.L. Speyer. Fault detection and identification for advanced vehicle control systems. In *Proceedings of the 13th World Congress, International Federation of Automatic Control*, volume Q, pages 201–206, 1997.

[5] A. Duyar and W. Merrill. Fault diagnosis for the space shuttle main engine. *Journal of Guidance, Control and Dynamics*, 15:384–389, March-April 1992.

[6] Puneet Goel, S.I. Roumeliotis, and G.S. Sukhatme. Robust localization using relative and absolute position estimates. In *Proceedings of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 1999.

[7] R.E. Kalman. A new approach to linear filtering and prediction problems. *ASME Journal of Basic Engineering*, 86:35–45, 1960.

[8] D.T. Magill. Optimal adaptive estimation of sampled stochastic processes. *IEEE Transactions on Automatic Control*, AC-10(4):434–439, 1985.

[9] P.S. Maybeck and P.D. Hanlon. Performance enhancement of a multiple model adaptive estimator. *IEEE Transactions on Aerospace and Electronic Systems*, 31(4):1240–1253, October 1995.

[10] P.S. Maybeck and D.L. Pagoda. Multiple model adaptive controller for the stol f-15 with sensor/actuator failures. In *Proceedings of the 20th Conference on Decision and Control*, pages 1566–1572, December 1989.

[11] T.R. Menke and P.S. Maybeck. Sensor/actuator failure detection in the vista f-16 by mutiple model adaptive estimation. *IEEE Transactions on Aerospace and Electronic Systems.*, 31(4):1218–1229, October 1995.

[12] S.I. Roumeliotis, G.S. Sukhatme, and G.A. Bekey. Fault detection and identification in a mobile robot using multiple-model estimation. In *Proceedings of the 1998 IEEE International Conference in Robotics and Automation*, May 1998.

[13] S.I. Roumeliotis, G.S. Sukhatme, and G.A. Bekey. Sensor fault detection and identification in a mobile robot. In *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1998.

[14] S.I. Roumeliotis, G.S. Sukhatme, and G.A. Bekey. Smoother based 3d attitude estimation for mobile robot localization. Technical report, University of Southern California, August 1998.