

Real-Time Model-Based Fault Detection and Isolation for UGVs

A. Monteriù · P. Asthana · K. P. Valavanis · S. Longhi

Received: 20 January 2009 / Accepted: 2 March 2009 / Published online: 26 March 2009
© Springer Science + Business Media B.V. 2009

Abstract The paper presents a model-based sensor fault detection and isolation system applied in real-time to unmanned ground vehicles. Structural analysis is applied on the nonlinear model of the vehicle for building the residual generation module, followed by an ad-hoc residual evaluation module for detecting single and multiple sensor faults. The overall proposed diagnosis scheme has been tested in real-time on a real mobile robot in an outdoors environment and for different tasks. The obtained experimental results are satisfactory in terms of diagnosis performance and real-time implementation.

Keywords Fault detection · Fault diagnosis · Real-time systems

1 Introduction

This paper has been motivated by the challenge of deriving a model-based sensor Fault Detection and Isolation (FDI) system to be implemented and tested in real-time on an Unmanned Ground Vehicle (UGV). The conceptual structure of the

A. Monteriù (✉) · S. Longhi
Dipartimento di Ingegneria Informatica, Gestionale e dell'Automazione,
Università Politecnica delle Marche, Via Breccie Bianche, 60131 Ancona, Italy
e-mail: a.monteriu@univpm.it

S. Longhi
e-mail: sauro.longhi@univpm.it

P. Asthana · K. P. Valavanis
Department of Computer Science and Engineering, University of South Florida,
Tampa, FL 33620, USA

P. Asthana
e-mail: pasthana@cse.usf.edu

K. P. Valavanis
e-mail: kvalavan@cse.usf.edu

derived sensor FDI system comprises the two main stages [1] widely accepted by the fault diagnosis community: the residual generation and the residual evaluation. The residual generator module is developed using the “structural analysis” [2–7]. The developed ad-hoc residual evaluation solution consists on an adaptive/moving threshold decision module where probability density functions of residuals are not required.

In this application study, the considered UGV is the differential drive vehicle ATRV-Jr (All Terrain Robot Vehicle Junior), equipped with a Global Positioning System (GPS), an Inertial Measurement Unit (IMU), and two incremental optical encoders mounted on driving wheels. In order to improve residual sensitivity, a Kalman Filter (KF) based error model has been introduced for the vehicle IMU sensor to estimate true values of robot orientation, angular rate, linear acceleration, velocity, position and related errors [8]. Experimental validation and verification is provided in terms of exhaustive tests in real-time considering single and multiple sensor faults. Additive sensor faults are considered here describing changes in the system states interpreted as sensor faults.

The main contribution of this work is the real-time application of the considered model-based sensor fault detection and isolation scheme and its integration with a background distributed architecture that supports the navigation of heterogeneous robot systems [9]. The paper is organized as follows. Section 2 summarizes the model of the robot. Section 3 presents the derived model-based fault detection and isolation system, explaining both the residual generation and evaluation modules. Experimental results are shown and detailed in Section 4, while Section 5 ends the paper.

2 The UGV Model

The differential drive UGV is assumed to move in an area where the altitude $h(t)$ is assumed constant. For such a vehicle, the North-East (NE) tangent plane [10] is considered, with the X -axis coinciding with the N -axis and, the E -direction coinciding with the Y -axis, respectively. In this frame the UGV orientation $\psi(t)$ describes the angle between the main axis of the robot and the X -axis. Localization of the UGV in this frame requires knowledge of coordinates x and y of the midpoint between the two driving wheels and of the angle $\psi(t)$. The kinematics equations have the form [11]:

$$\dot{x}(t) = \dot{f}(t) \cos \psi(t) \quad (1)$$

$$\dot{y}(t) = \dot{f}(t) \sin \psi(t) \quad (2)$$

$$\dot{\psi}(t) = r(t) \quad (3)$$

where $\dot{f}(t)$ and $r(t)$ are, respectively, the forward and angular velocities of the robot, expressed by:

$$\dot{\psi}(t) = \frac{D}{2L} (\omega_R(t) - \omega_L(t)) \quad (4)$$

$$\dot{f}(t) = \frac{D}{4} (\omega_R(t) + \omega_L(t)) \quad (5)$$

where $\omega_R(t)$ and $\omega_L(t)$ are the angular velocities of the right and left wheels, respectively, D is the wheel diameter and L is the distance between the wheels.

The robot localization system uses two incremental optical encoders mounted on driving wheel that provide right and left wheel angle velocities $\omega_R(t)$ and $\omega_L(t)$, respectively; an Inertial Measurement Unit that provides the forward linear acceleration $\ddot{f}(t)$ and the angular velocity $r(t)$ of the vehicle; a Global Positioning System antenna measuring the latitude $\lambda(t)$, the longitude $\Phi(t)$ and altitude $h(t)$ with respect to the Earth-Centered-Earth-Fixed (ECEF) frame. The last set of measurements is related to the vehicle variables by the set of the following equations:

$$\dot{x}(t) = \dot{\lambda}(t) R_{\lambda_0} \tag{6}$$

$$\dot{y}(t) = \dot{\Phi}(t) R_{\Phi_0} \cos \lambda(t) \tag{7}$$

and the constants R_{λ_0} and R_{Φ_0} depend on the altitude $h(t)$, on the radius of curvature and on the transverse radius of curvature, which are all assumed constant for the developed experiments without loss of generality.

All parameters of the UGV model are considered known, without any uncertainty.

3 Proposed Sensor Fault Detection and Isolation System

3.1 Structural Analysis: the Residual Generation Module

The nonlinear UGV model is considered as a set of nine constraints applied to a set of unknown variables $\{\psi, \dot{\lambda}, \dot{\Phi}, \dot{f}\}$ and to the set of known variables $\{\lambda, \Phi, \ddot{f}, r, \dot{\psi}, \omega_L, \omega_R\}$ where the time dependency has been omitted for simplicity. The matching algorithm identifies [3, 12] the set of matched constraints, and back tracking procedure gives the residuals r_1, r_2, r_3 and r_4 :

$$r_1 = \int_0^t \left(R_{\lambda_0} \frac{d\lambda}{d\tau} - \left(\frac{D}{2} \omega_R - \frac{L}{2} r \right) \cdot \cos \left(\arcsin \left(\frac{d\Phi}{d\tau} \cdot \frac{2R_{\Phi_0} \cos(\lambda)}{D\omega_R - Lr} \right) \right) \right) d\tau \implies [r_1] = [m] \tag{8}$$

$$r_2 = \ddot{f} - \frac{d}{dt} \left(\frac{D}{2} \omega_R - \frac{L}{2} r \right) \implies [r_2] = \left[\frac{m}{s^2} \right] \tag{9}$$

$$r_3 = r - \frac{d}{dt} \left(\arcsin \left(\frac{d\Phi}{dt} \cdot \frac{2R_{\Phi_0} \cos(\lambda)}{D\omega_R - Lr} \right) \right) \implies [r_3] = \left[\frac{rad}{s} \right] \tag{10}$$

$$r_4 = \omega_L - \omega_R + \frac{2L}{D} r \implies [r_4] = \left[\frac{rad}{s} \right] \tag{11}$$

In order to have a strong detectability of the faults, the second residual has been integrated and added to the residual set as an extra one:

$$r_5 = \int_0^t r_2(\tau) d\tau \implies [r_5] = [rad] \tag{12}$$

All residuals are discretized for on-line implementation.

3.2 Residual Evaluation Module

The most simple and straightforward method for fault decision consists in a threshold test of the residual. One way of designing thresholds is to use statistical methods (see e.g. [13] and [14]), fuzzy decision logic (see e.g. [15]) or neural networks. However, if constant thresholds are used one has to cope with the problem of the effects of unknown inputs including modelling errors and disturbances. This means, if the threshold is chosen too small, false alarms occur, if the threshold is chosen too large, small faults can not be detected. The effect of the modelling errors and disturbances depends on the operating conditions of the process. From the results in e.g. [16] and [17], it can be concluded that a time varying robust threshold which is adapted to the operating conditions of the process, i.e. a dynamic threshold, is better than a constant. In [18, 19], robust thresholds are suggested for the special case of full state measurement systems. In this paper, an algorithm is derived on how to design the upper and lower bounds design of the dynamic robust threshold, such that the threshold will fulfill the requirement that no false alarm is produced.

Let $\{r_i(kT_s) : k \in [(j-1) \cdot n + 1, j \cdot n]\}$ be an observed sequence of the i -th residual $r_i(kT_s)$ in a generic j -th sliding window of size n (number of readings in the sliding window). Before the unknown change time k_0T_s , the mean value θ of the residual is constant and equal to θ_0 . After a change the parameter is equal to θ_1 . The hypotheses are:

$$\mathcal{H}_0 : \theta = \theta_0 \text{ for } ((j-1) \cdot n + 1) \leq k \leq j \cdot n$$

$$\mathcal{H}_1 : \theta = \theta_0 \text{ for } ((j-1) \cdot n + 1) \leq k \leq k_0 - 1 \text{ and } \theta = \theta_1 \text{ for } k_0 \leq k \leq j \cdot n$$

The on-line problem is to detect the occurrence of the change as soon as possible. The *decision test* is based on the following:

$$\text{if } r_i(kT_s) < H_{i,j} \text{ or } r_i(kT_s) > h_{i,j} \text{ accept } \mathcal{H}_0 \text{ and set } g_i(kT_s) = 0$$

$$\text{if } r_i(kT_s) \geq H_{i,j} \text{ accept } \mathcal{H}_1 \text{ and set } g_i(kT_s) = +1$$

$$\text{if } r_i(kT_s) \leq h_{i,j} \text{ accept } \mathcal{H}_1 \text{ and set } g_i(kT_s) = -1$$

where $g_i(kT_s)$ is the decision function of the i -th residual, $H_{i,j}$ and $h_{i,j}$ are upper and lower thresholds, respectively (to detect deviations from θ_0 in both directions). For each specific residual, $H_{i,j}$ and $h_{i,j}$ are properly chosen. In particular, statistical investigation of fault free residuals has resulted in fixing the thresholds of the first and fifth residual as shown in Table 1 (different experimental tests have validated this choice).

However, because of noise, it is not possible to set constant values for $H_{i,j}$ and $h_{i,j}$ for the remaining three residuals. In order to set threshold values, an ad-hoc

Table 1 Threshold values for the first and last residual

Residual	$H_{i,j}$	$h_{i,j}$
$r_1(kT_s)$	0.5 m	-0.5 m
$r_5(kT_s)$	0.042 °	-0.042 °

algorithm has been applied to the i -th residual $r_i(kT_s)$ with $i = 2, 3, 4$. This proposed algorithm is as follows:

- Let $j = 1$. In the first sliding window $j = 1$ of size n , consider that the system operates in a fault free working mode, that is, \mathcal{H}_0 is accepted. Determine the absolute minimum $m_{i,j}$ and absolute maximum $M_{i,j}$ values of the observed sequence of the i -th residual $r_i(kT_s)$. Set upper and lower thresholds for the next sliding window as follows:

$$H_{i,j+1} = M_{i,j} + \Delta_{i,j}^+ \tag{13}$$

$$h_{i,j+1} = m_{i,j} - \Delta_{i,j}^- \tag{14}$$

with $\Delta_{i,j}^+$ and $\Delta_{i,j}^-$ constant values resulting from experimental investigations of faultless operation residuals.

- Let $j = j + 1$. In the sliding window j of size n at each sample time $k \in [(j - 1) \cdot n + 1, j \cdot n]$, the *decision test* runs on-line and the decision function $g_i(kT_s)$ of the i -th residual is updated:

if \mathcal{H}_0 is accepted, set $g_i(kT_s) = 0$

if \mathcal{H}_1 is accepted, set $g_i(kT_s) = +1$ or set $g_i(kT_s) = -1$

If \mathcal{H}_0 is accepted, determine the absolute minimum $m_{i,j}$ and absolute maximum $M_{i,j}$ of the i -th residual $r_i(kT_s)$ in the considered sliding window. Set the upper and lower thresholds for the next sliding window as in Eqs. 13 and 14, respectively. Otherwise, if \mathcal{H}_1 is accepted, and this could occur also for one sample time of the sliding window, then $H_{i,j+1}$ and $h_{i,j+1}$ keep the predetermined values, i.e. $H_{i,j+1} = H_{i,j}$, $h_{i,j+1} = h_{i,j}$.

- Go to step 2).

Due to the stochastic framework and considering the probability density functions of the residuals as unknown, it has been chosen to characterize the thresholds variations, $\Delta_{i,j}^+$ and $\Delta_{i,j}^-$, through experimental trials [20]. Firstly, faultless residuals have been recorded over long periods of time (20 h), determining the absolute maximum and minimum of each residual. Dividing these long periods in small time windows, it has been possible to determine the maximum and the minimum value of the residual in each time window. Secondly, the variations among the maximum values and then among the minimum values of all time windows have been compared, permitting to characterize the thresholds variation.

Experimentation resulted in $\Delta_{i,j}^+$ and $\Delta_{i,j}^-$ values as shown in Table 2.

Note that, in the j -th sliding window, if \mathcal{H}_0 is accepted, then upper and lower thresholds are based on maximum and minimum values of $r_i(kT_s)$, with $i = 2, 3, 4$, determined in the previous ($(j - 1)$ -th) sliding window, otherwise they keep the previous values. This allows to update thresholds in every new sliding window. The

Table 2 Delta values for the second, third and fourth residual

Residual	$\Delta_{i,j}^+$	$\Delta_{i,j}^-$
$r_2(kT_s)$	$5 M_{2,j} $	$5 m_{2,j} $
$r_3(kT_s)$	$ M_{3,j} + \frac{ m_{3,j} }{2}$	$ M_{3,j} + \frac{ m_{3,j} }{2}$
$r_4(kT_s)$	$2 M_{4,j} $	$2 m_{4,j} $

thresholds are independent from the occurrence of the faults, permitting to detect them. This algorithm has been validated through a large set of experimental tests, as discussed in the following section.

4 Experimental Results

4.1 Background Information

The proposed FDI system has been implemented and tested experimentally on a differential drive ATRV-Jr mobile robot platform. All experiments have been performed outdoors in an environment with several tall buildings (affecting GPS readings), vegetation and palm trees. The robot sensor suite includes a color camera mounted on a pan/tilt mechanism, Sick planar laser range finder, electronic compass, Garmin 16A GPS, odometers, wireless Ethernet connectivity and Crossbow's IMU 400CC-200, all connected to and integrated with the ATRV-Jr on-board computer (Pentium IV, 3.2 GHz, 2 GB Memory) through a Rocketport multi serial port card. Three sensors are used to evaluate the proposed system: Garmin 16A GPS, internal odometry and Crossbow's IMU 400CC-200, with measured resolutions as shown in Table 3 [21, 22]. For this kind of applications, the onboard IMU is the noisier sensor. In order to reduce the measurement noise, the IMU readings are preprocessed by a Kalman Filter [8], as confirmed by the low noise levels present in the experimental results reported in the next.

GPS, IMU and Odometry data are recorded in real-time as the robot follows the test trajectories. Recorded sensor data are used as input to generate on-line the five residuals which are the input of the change detection module. Additive sensor faults are considered as 'user imposed / software generated step faults' added to the recorded sensor data at different times as the robot moves along a trajectory. The magnitude of the step faults is shown in Table 4; these values have been estimated by considering sensor resolution and the signal to noise ratio of each residual. These choices have been confirmed by experimental trials.

An advantage of the implemented system in its current configuration is that it may function in 'actual' real-time and 'pseudo' real-time. The latter refers to collecting sensor data in real-time as the robot moves, but using them off-line to test and evaluate the sensor FDI system by generating residuals and change detectors. The diagnostic system has been integrated in the 'distributed field robot architecture' [9], used for navigation and control of ATRV-Jr.

4.2 Implementation and Integration Issues

In order to verify the effectiveness and also the limits of the proposed scheme, the implementation details and issues are introduced.

Table 3 Sensor resolution

Sensor	Resolution
IMU 400C-200 gyroscope	< 0.05 °/s
IMU 400C-200 accelerometer	< 1.25 mg
Garmin 16A GPS	< 0.02 m

Table 4 Magnitude of the experimental step faults

Parameter	Fault magnitude
GPS	1.25 m
Forward acceleration	12.5 mg
Yaw angular rate	0.17 °/s
Left wheel angular velocity	2.85 °/s
Right wheel angular velocity	2.85 °/s

The distributed field robot architecture [9] extends the ‘sensor fusion effects’ managerial architecture [23] and it is developed in Java/Jini using modular services to implement robot capabilities, including sensors, effectors, and behaviors. Modules are exported to a distributed run-time system as services with certain attributes and types. Services can then be searched for (using a distributed-object lookup service) based on functional attributes rather than details of actual implementation or physical location. This architecture allows a decoupling of client and server, providing an interface (proxy) to the requesting process in a modular fashion regardless of where the requested service physically resides or how it is implemented at the local level [24]. The distributed field robot architecture implements ‘*Sensing Manager*’ module that is responsible for error classification and handling.

The operating system used is Red Hat Linux 9, kernel version 2.6.7. The distributed field robot architecture uses the standard Java Virtual Machine (JVM) thread scheduling mechanism to handle various service threads within a JVM. The specific low level ‘distributed field robot architecture’ services utilized in this research are: GPS service to provide the latitude and longitude of the UGV; IMU service to provide the acceleration and angular rate of the UGV; Odometry Service to provide UGV position; Drive Motor effector service to drive the UGV along a specific trajectory.

All ATRV-Jr controller designs and navigation routines have been derived using MATLAB/Simulink as well as Java. The MATLAB workspace environment is wrapped with JMatLink in conjunction with the Jini distributed object platform allowing modules and services implemented as native interpreted MATLAB code to be accessed as remote and distributed objects and be directly incorporated into behavioral architectures. This configuration provides also a complete MATLAB based simulation environment that may be used to test and validate off-line the sensor FDI system using real sensor data (collected in real-time but processed off-line).

Two modules are developed in Java: the ‘*ResidualCalcOffline*’ and the ‘*ResidualCalcOnline*’. The ‘*ResidualCalcOffline*’ module receives real-time GPS, IMU and Odometry data and generates residuals and change detectors off-line duplicating in reality what the MATLAB/Simulink module does. This module is also being converted to a Java based ‘*ResidualCalcOnline*’ one that generates on-line residuals and change detectors during actual experiments. ‘*ResidualCalcOnline*’ updates the ‘*Sensing Manager*’ with the sensor status. ‘*Sensing Manager*’ passes on the sensor status to a client graphical user interface. Whenever a sensor fault occurs, the ‘*Sensing Manager*’ updates the services that are using the faulty sensor and disables the faulty sensor. ‘*Sensing Manager*’ enables the faulty sensors when they recover from the fault.

The residuals have been numerically computed in real-time making use of numeric methods; specifically, trapezoidal rule has been used as numerical integration, and

forward Euler method has been used as numerical differentiation. Due to the accuracy of adopted numerical methods, each residual computation has been reinitialized in each sliding window, permitting to bound the unavoidable numerical errors.

4.3 Real-Time Issues

One data sample of '*ResidualCalcOnline*' consists of an IMU, odometry and GPS readings recorded at the same time. '*ResidualCalcOnline*' includes two threads: i) '*Data collection*' thread that records continuously readings from sensor devices at a (best effort) sampling rate of 200 ms; this sample time interval is imposed by the slow GPS update rate; ii) '*Residual calculation*' thread that calculates residuals and change detectors on-line by retrieving data stored by the '*Data collection*' thread. The states of '*Data collection*' are: *Executing*—the thread collects and records sensor readings; *Waiting*—the thread sleeps for the 200 ms time interval; *Ready*—the thread is ready to collect sensor data and waits for CPU allocation. The states of '*Residual calculation*' are: *Executing*—the thread calculates residuals and change detectors for a particular data sample; *Waiting*—the thread is waiting for new data samples to be recorded; *Ready*—the thread is ready to calculate residuals and change detectors for the new data samples and is waiting for CPU allocation. However, close observation of sensor readings reveals that the 200 ms sample time is not accurately achieved and that for each data sample there is a small time lag between the three sensor readings. This happens because: i) the three sensor readings cannot be recorded at the same time, and a data collection order is considered: IMU, odometry and GPS, introducing a small time lag between the IMU–odometry and odometry–GPS readings; ii) the Java garbage collector obstructs distributed field robot architecture from gaining complete control over the CPU scheduling time. Therefore, hard real-time guarantees cannot be achieved.

Performed experiments have resulted in the sequence diagram and variations reported in Fig. 1. From Fig. 2, the average of time interval between data collection and its corresponding residual and change detector generation is of 0.05 s with single high delays of 0.1 s, that are acceptable for the considered application.

4.4 Results Discussion

Sensor data are collected and residuals are generated on-line. In order to bound the error propagation, numeric integration and derivation for residuals are made inside a small and finite time window (max 10 time samples), and after this time, the numeric integrative and derivative process have been reinitialized and restarted. This solution has been proposed as a tradeoff between accuracy and computational efforts. The preprocessing of the IMU readings, performed by a Kalman filter, makes this a feasible solution, as confirmed by experimental results. For cases with one sensor fault, after a fixed number of data samples, an additive fault is introduced into one of the sensors. Experiments consider single and multiple faults per trial following the same principle of introducing additive faults.

The implemented diagnostic system is able to detect every single/multiple faults of the considered sensor equipment. Moreover, sensor faults are detected and isolated in all situations where a single sensor fault has occurred at a time. The situation is slightly different when multiple sensor faults occur simultaneously. In this work,

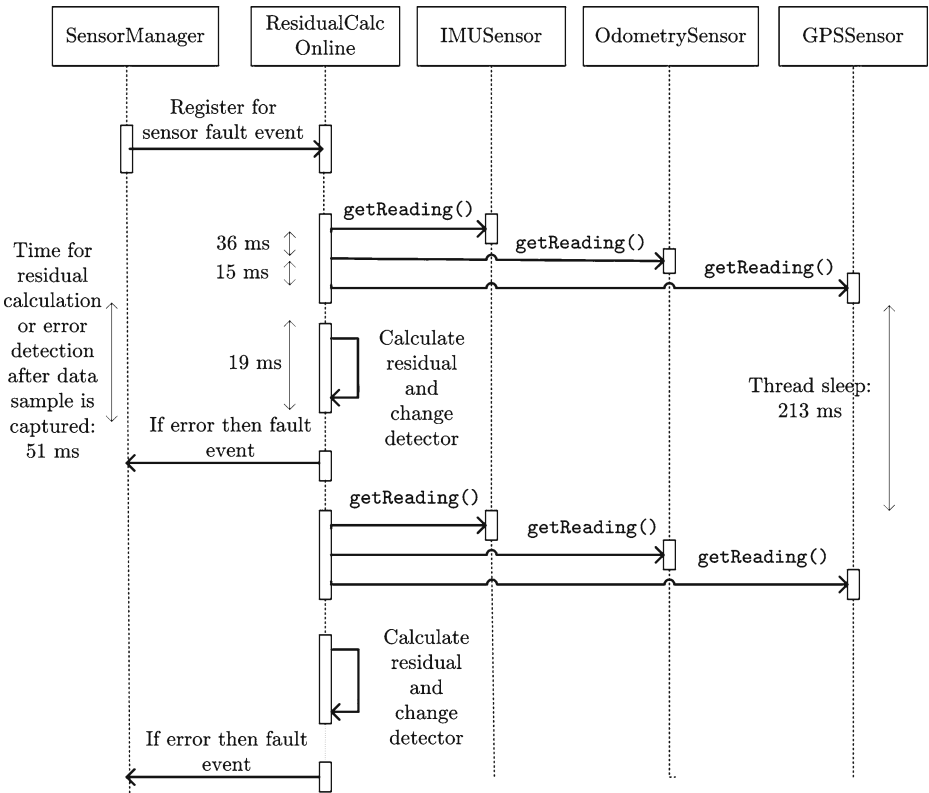


Fig. 1 Sequence diagram for service interaction. The times (in milli seconds) above reflect the observed time delays

the isolation strategy for multiple faults assumes that a strong detectable sensor fault always makes certain residuals exceed their thresholds; in literature this is known as ‘exoneration assumption’ [25, 26] which is the assumption by default in

Fig. 2 The time interval between a data sample getting recorded and its corresponding residual and change detector generation for different data samples of a developed trajectory experiment

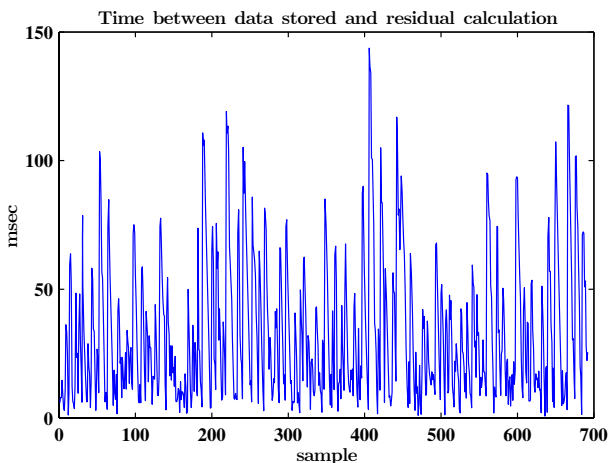


Table 5 Effects of the sensor faults on the residuals

\nearrow	f_{GPS}	f_{ACC}	f_{GYRO}	f_{LE}	f_{RE}
r_1	1	0	0	0	0
r_2	0	1	0	0	0
r_3	0	0	1	0	0
r_4	0	0	1	1	1
r_5	0	1	0	0	1

the FDI approach. In multiple fault case, fault isolability cannot be guaranteed in all situations, as it is resumed in the fault signature table (Table 5). In this table, f_{GPS} , f_{ACC} , f_{GYRO} , f_{LE} and f_{RE} denote “GPS fault”, “accelerometer fault”, “gyroscope fault”, “left optical encoder fault” and “right optical encoder fault”, respectively, while “1” (“0”) indicates that the fault in the corresponding column affects (does not affect) the residual of the corresponding row. For instance, a GPS fault affects only the first residual r_1 and not the other residuals, as indicated by “1” in the first row of this table. A gyro fault affects both residuals r_3 and r_4 , as indicated in the third and fourth rows of the table. It is important to notice that all sensor faults reported in Table 5, are those which are strongly detectable [3].

Table 5 shows that accelerometer fault affects both residuals r_2 and r_5 and this is also validated by experimental results shown in Fig. 3 for a robot linear trajectory. In this case, the accelerometer fault was occurred at sample time 380 (see Fig. 3) and the change detector of the residual issued an alarm after one sample time; this is the minimum possible detection time for the considered system architecture.

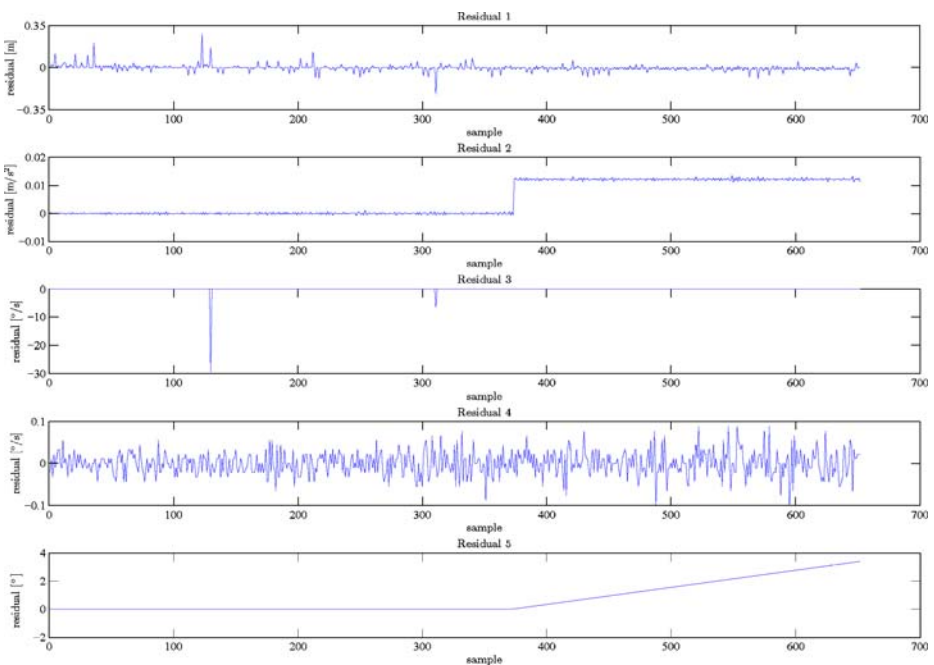


Fig. 3 Residuals in ACC faulty situation

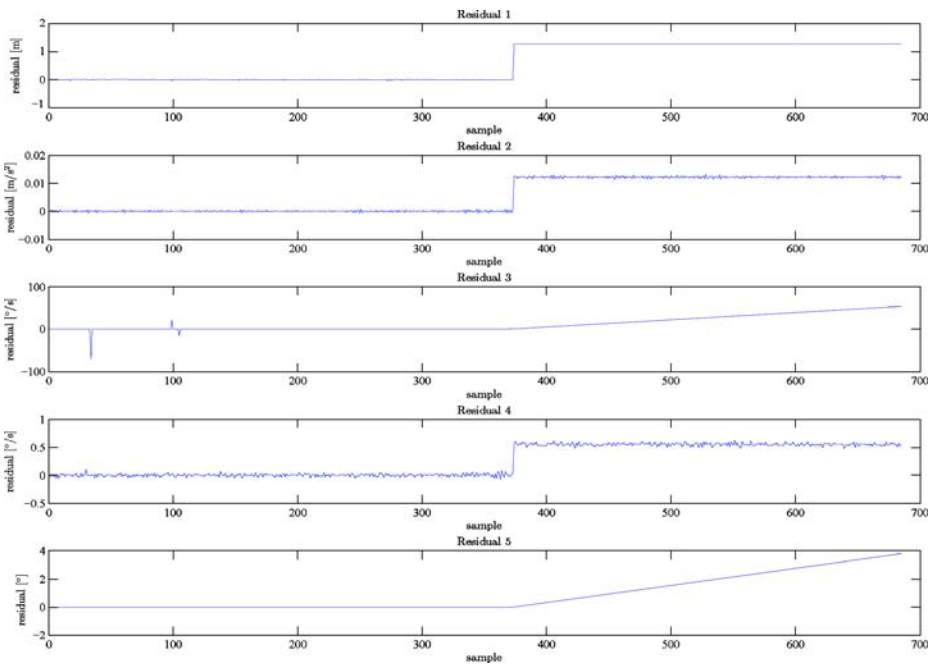


Fig. 4 Residuals in GPS + ACC + GYRO faulty situation

Similar results have been obtained in many other real-time performed experiments, which are not reported here for a sake of brevity. The experimental result for multiple sensor faults related to following a straight line trajectory is presented in Fig. 4. This figure illustrates the situation when GPS, accelerometer and gyroscope fail simultaneously, where all five residuals change their mean values. In this case, it is possible to assert that GPS, accelerometer and gyroscope fail for sure (r_1 , r_2 and r_3 deviate from zero), but nothing may be said about the faulty condition of left and/or right optical encoder, as indicated in Table 5. In all other cases, combining the information provided by all five residuals, it is guaranteed to distinguish and isolate the faults, but not in the case mentioned above. This limitation can be overcome, for instance, by adding new sensors so that redundancy increases. Note that also actual faults, such as the wheel slippage, have been considered in different experimental

Table 6 Two simultaneous faults

	GPS	ACC	GYRO	LE	RE
GPS+ACC	D	D	-	-	-
GPS+GYRO	D	-	D	-	-
GPS+LE	D	-	-	D	-
GPS+RE	D	-	-	-	D
ACC+GYRO	-	D	D	-	-
ACC+LE	-	D	-	D	-
ACC+RE	-	D	-	-	D
GYRO+LE	-	-	D	U	-
GYRO+RE	-	-	D	-	D
LE+RE	-	-	-	U	-

Table 7 Three simultaneous faults

\nearrow	GPS	ACC	GYRO	LE	RE
GPS+ACC+GYRO	D	D	D	–	–
GPS+ACC+LE	D	D	–	D	–
GPS+ACC+RE	D	D	–	–	D
GPS+GYRO+LE	D	–	D	–	U
GPS+GYRO+RE	D	–	D	–	D
GPS+LE+RE	D	–	–	U	D
ACC+GYRO+LE	–	D	D	U	–
ACC+GYRO+RE	–	D	D	–	U
ACC+LE+RE	–	D	–	U	U
GYRO+LE+RE	–	–	D	U	U

trials. Although the wheel slippage has been neglected for developing the nonlinear UGV model, this is treated as a fault on the left or right encoder by the proposed diagnostic system. In effect, slippage of a wheel means loss of sensor measurement and this affects residuals r_4 and/or r_5 , thus the diagnostic system is able to issue an alarm. All experimental results are summarized in Tables 6, 7 and 8. It is shown that the proposed approach results in overall excellent performance in all experimental tests. In these tables, “D” indicates that the sensor fault is detected, while “U” indicates that the sensor fault is undetermined.

4.5 Comparative Analysis

Different solutions have been proposed for solving the residual change detection problem (see e.g. [13, 27–30]). In the comparative analysis, the Particle Filter [28, 31–33] is considered that is an appropriate tool for the application considered in this work, where no Gaussian hypothesis are formulated. The probability density function of each residual is estimated by particle filter that is a sequential Monte Carlo filter where the complete posterior distribution of the estimates are represented through samples or particles. Therefore, the central idea of the considered approach is to compute the joint likelihood of the observations conditioned on each hypothesized model through Monte-Carlo estimation that uses the complete sample-based probability density function information provided by the particle filter [34–36].

Performance of the ad-hoc residual evaluation module has been compared with this classical solution based on the particle filter. Experimental tests have shown that the proposed ad-hoc solution has the same fault detection performances. A sample of the developed comparative analysis is reported in Table 9, related to a GPS fault experiment. GPS fault has been detected at the same sample time. The same results have been obtained for all other faulty experiments. However, the decision module based on the particle filter implementation requires more computational time,

Table 8 Four simultaneous faults

\nearrow	GPS	ACC	GYRO	LE	RE
GPS+ACC+GYRO+LE	D	D	D	U	–
GPS+ACC+GYRO+RE	D	D	D	–	U
GPS+ACC+LE+RE	D	D	–	U	U
GPS+GYRO+LE+RE	D	–	D	U	D
ACC+GYRO+LE+RE	–	D	D	U	U

Table 9 Comparison between the ad-hoc solution and the PF based solution

	GPS fault time (samples)	PF based (samples)	Ad-hoc (samples)	PF calc time	Ad-hoc calc time
Test#1	170	171	171	9.7 ms	8.4 ms
Test#2	210	211	211	9.9 ms	8.7 ms
Test#3	380	381	381	9.5 ms	7.9 ms

therefore, the proposed adaptive/moving thresholds decision module is preferable to use for on-line applications. Although the proposed evaluation module has been developed ad-hoc for this application, it can be particularized and applied in several real-time applications in a non-gaussian framework. In fact, use of this algorithm requires only a set of fault-free data samples collected in one time window. This training set is sufficient to initialize and run the developed algorithm.

5 Conclusions

The paper has presented a model-based sensor fault detection and isolation scheme that has been implemented and tested on a mobile robot platform. Implementation details of the residual generation and evaluation modules have been presented with details related to structural analysis to identify simultaneously occurring multiple sensor faults through five residuals. A Kalman filter is considered for improving fault detection sensitivity. An adaptive/moving thresholds decision module is derived and compared to the particle filter based decision module. The reduced computational efforts required by the adaptive/moving threshold decision module permits to use this module for real-time applications.

Another significant aspect and contribution of this paper is the on-line experimentation of the diagnostic system and its ‘dual’ validation in actual real-time and pseudo real-time through MATLAB and JAVA development environments. All sensor faults have been detected and isolated with a maximum detection delay of one sample time. Exhaustive experiments suggest that the proposed scheme is reliable and robust for the considered prototype platform ATRV-Jr. Thus, the developed results can be easily extended to different slowing moving platforms.

Current research relates to improved accuracy, reliability and robustness of the proposed diagnostic system. A possible solution under investigation is to increase the sensitivity of the residuals as much as possible. Further, although abrupt faults have been considered in this paper, current research studies incipient and time varying faults. Different residual generation and evaluation modules are being tested to accommodate such types of faults.

References

1. Chow, E.Y., Willsky, A.S.: Issues in the development of a general design algorithm for reliable failure detection. In: Proc. 19th IEEE Conf. Decis. and Contr., pp. 1006–1012, Albuquerque, 10–12 December 1980
2. Staroswiecki, M., Declerck, P.: Analytical redundancy in non linear interconnected systems by means of structural analysis. In: IFAC Advanced Information Processing in Automatic Control (1989)

3. Blanke, M., Kinnaert, M., Lunze, J., Staroswiecki, M.: *Diagnosis and Fault-Tolerant Control*, 2nd edn. Springer, Heidelberg (2006)
4. Blanke, M., Niemann, H., Lorentzen, T.: Structural analysis—a case study of the Rømer satellite. In: *Proc. of IFAC Safeprocess 2003*, Washington, DC, 9–11 June 2003
5. Krysander, M.: *Design and analysis of diagnostic systems utilizing structural methods*. Department of Electrical Engineering, Linköpings Universitet (2003)
6. Pulido, B., Gonzalez, C.: Possible conflicts: a compilation technique for consistency-based diagnosis. *IEEE Trans. Syst. Man Cybern. Part B* **34**(5), 2192–2206 (2004)
7. Düstegör, D., Frisk, E., Cocquempot, V., Krysander, M., Staroswiecki, M.: Structural analysis of fault isolability in the damadics benchmark. *Control Eng. Pract.* **14**(6), 597–608 (2006)
8. Barshan, B., Durrant-Whyte, H.F.: Inertial navigation systems for mobile robots. *IEEE Trans. Robot. Autom.* **11**(3), 328–342 (1995)
9. Long, M.: *Creating a Distributed Field Robot Architecture for Multiple Robots*. Master's thesis, University of South Florida (2004)
10. Farrell, J.A., Barth, M.: *The Global Positioning System & Inertial Navigation*. McGraw-Hill, New York (1998)
11. Wang, C.M.: Localization estimation and uncertainty analysis for mobile robots. In: *IEEE International Conference on Robotics and Automation*, pp. 347–351, Philadelphia, April 1988
12. Dulmage, A., Mendelsohn, N.: Coverings of bipartite graphs. *Can. J. Math.* **10**, 517–534 (1958)
13. Basseville, M., Nikiforov, I.V.: *Detection of Abrupt Changes: Theory and Application*. Prentice-Hall, Englewood Cliffs (1993)
14. Gustafsson, F.: *Adaptive Filtering and Change Detection*. Wiley, New York (2000)
15. Frank, P.: Residual evaluation for fault diagnosis based on adaptive fuzzy thresholds. In: *IEE Colloquium on Qualitative and Quantitative Modelling Methods for Fault Diagnosis*, p. 4, April 1995
16. Bask, M., Johansson, A.: Model-based supervision of valves in a flotation process. In: *Proceedings of the 42nd IEEE Conference on Decision and Control*, 2003, vol. 1, pp. 744–749. IEEE, Piscataway (2003)
17. Johansson, A., Bask, M., Norlander, T.: Dynamic threshold generators for robust fault detection in linear systems with parameter uncertainty. *Automatica* **42**(7), 1095–1106 (2006)
18. Zhang, X., Polycarpou, M., Parisini, T., Inc, I., Rockville, M.: A robust detection and isolation scheme for abrupt and incipient faults in nonlinear systems. *IEEE Trans. Automat. Contr.* **47**(4), 576–593 (2002)
19. Zhang, X., Parisini, T., Polycarpou, M.: Sensor bias fault isolation in a class of nonlinear systems. *IEEE Trans. Automat. Contr.* **50**(3), 370–376 (2005)
20. Patton, R., Frank, P., Clarke, R.: *Fault Diagnosis in Dynamic Systems: Theory and Application*. Prentice-Hall, Upper Saddle River (1989)
21. XBOW: Crossbow homepage. <http://www.xbow.com/index.aspx> (2008)
22. GARMIN: GARMIN manual. <http://www.garmin.com/support/usermanual.jsp> (2009)
23. Murphy, R.: *Intro to AI Robotics*. MIT, Cambridge (2000)
24. Long, M., Gage, A., Murphy, R., Valavanis, K.: Application of the distributed field robot architecture to a simulated demining task. In: *IEEE International Conference on Robotics and Automation (ICRA)*, Barcelona, 18–22 April 2005
25. Cordier, M., Dague, P., Dumas, M., Levy, F., Montmain, J., Staroswiecki, M., Travé-Massuyès, L.: A comparative analysis of ai and control theory approaches to model-based diagnosis. In: *14th European Conference on Artificial Intelligence*, pp. 136–140, Berlin, 20–25 August 2000
26. Celse, B., Cauvin, S., Heim, B., Gentil, S., Travé-Massuyès, L.: Model based diagnostic module for a fcc pilot plant. *Oil Gas Sci. Technol.* **60**(4), 661 (2005)
27. Gustafsson, F.: The marginalized likelihood ratio test for detecting abrupt changes. *IEEE Trans. Automat. Contr.* **41**, 66–78 (1996)
28. Kadiramanathan, V., Li, P., Jaward, M., Fabri, S.: Particle filtering-based fault detection in nonlinear stochastic systems. *Int. J. Syst. Sci.* **33**(4), 259–265 (2002)
29. Punsakaya, E., Andrieu, C., Doucet, A., Fitzgerald, W.J.: Bayesian curve fitting with applications to signal segmentation. *IEEE Trans. Signal Process.* **50**(3), 747–758 (2002)
30. Tourneret, J.Y., Doisy, M., Lavielle, M.: Bayesian retrospective detection of multiple change-points corrupted by multiplicative noise. Application to sar image edge detection. *IEEE Trans. Signal Process.* **83**(9), 1871–1887 (2003)
31. Kadiramanathan, V., Li, P., Jaward, M., Fabri, S.: A sequential Monte Carlo filtering approach to fault detection and isolation in nonlinear systems. In: *Proceedings of the 39th IEEE Conference on Decision and Control*, 2000, vol. 5. IEEE, Piscataway (2000)

32. Dearden, R., Clancy, D.: Particle filters for real-time fault detection in planetary rovers. In: Proceedings of the Thirteenth International Workshop on Principles of Diagnosis, pp. 1–6, Semmering, 2–4 May 2002
33. Verma, V., Gordon, G., Simmons, R., Thrun, S.: Real-time fault diagnosis [robot fault diagnosis]. *IEEE Robot. Autom. Mag.* **11**(2), 56–66 (2004)
34. Gordon, N.J., Salmond, D.J., Smith, A.F.M.: Novel approach to nonlinear/non-gaussian Bayesian state estimation. In: *Proc. Inst. Elect. Eng. F*, vol. 140, pp. 107–113 (1993)
35. Kitagawa, G.: Monte-carlo filter and smoother for non-gaussian nonlinear state space models. *J. Comput. Stat. Assoc.* **5**(1), 1–25 (1996)
36. Monteriu, A., Asthana, P., Valavanis, K., Longhi, S.: Model-based sensor fault detection and isolation system for unmanned ground vehicles: theoretical aspects (part i). In: 2007 IEEE International Conference on Robotics and Automation, pp. 2736–2743. IEEE, Piscataway (2007)