2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Model-Based Sensor Fault Detection and Isolation System for Unmanned Ground Vehicles: Experimental Validation (part II)

Andrea Monteriù*‡, Prateek Asthan*, Kimon Valavanis* and Sauro Longhi‡

*Department of Computer Science and Engineering
University of South Florida – Tampa, FL 33620

‡Dipartimento di Ingegneria Informatica, Gestionale e dell'Automazione
Università Politecnica delle Marche – 60131 Ancona, Italy

Abstract— **This paper presents implementation details of a model-based sensor fault detection and isolation system (SFDIS) applied to unmanned ground vehicles (UGVs). Structural analysis, applied to the nonlinear model of the UGV, is followed to build the residual generation module, followed by a residual evaluation module capable of detecting single and multiple sensor faults, as detailed in part I [1]. The overall proposed sensor fault detection and isolation system has been tested in real-time on the ATRV-Jr mobile robot when following different trajectories in an outdoors environment. The robot sensor suite includes a Global Positioning System (GPS) antenna, an Inertial Measurement Unit (IMU), and two incremental optical encoders.**

I. Introduction

THIS paper has been motivated by the challenge to implement on an unmanned ground vehicle (UGV) and test in real-time a model-based sensor fault detection and isolation (FDI) system by integrating a residual generation module with a residual evaluation module. Experimental validation of the proposed system and exhaustive tests are presented here in real-time considering single and multiple sensor faults. The UGV, in this case a differential drive ATRV-Jr, is equipped with a sensor suite that includes a Global Positioning System (GPS) antenna measuring absolute position in the geodetic coordinates, an Inertial Measurement Unit (IMU) measuring robot linear accelerations and angular velocities, and incremental optical encoders mounted on motors measuring motor rotation.

Given a UGV (or any system), faults may be overcome by using robust "model-based fault diagnosis". A model-based fault diagnosis system consists in principle of a residual generation module and a residual evaluation module [2]-[9] that evaluates residuals deciding about the likelihood and/or presence of a fault. The decision process/rule applied to determine if any faults have occurred may be a threshold test on the instantaneous values or moving averages of the residuals, or it may follow statistical decision theory techniques. A drawback of the model-based techniques, especially those applied to the nonlinear systems, is that these may be very sensitive to measurement noise, as it is the case with UGVs. Therefore, noise filtering of the sensor measurements has been introduced. A Kalman filter (KF) based error model has been derived for the vehicle IMU sensor to estimate true values of robot orientation, angular rate, linear acceleration, velocity, position and related errors; filtered values are then considered in the residual generation module as shown in Figure 1 of the first part of this work [1] (see also [10]). Note that the KF may be enhanced to include GPS and encoder data, but since experiments have been performed in a limited geographic area, filtered GPS data do not offer any substantial improvement of the performance of the developed fault detection and isolation system.

In this work, additive and abrupt sensor faults have been considered, describing changes in the system states interpreted as sensor faults. The residual generator module [11], generates specific residuals to detect sensor faults following structural analysis [12]-[15] of the UGV nonlinear model, determining existing redundancies in sensors.

The residual evaluation module reduces to the problem of detecting a change in the mean of a random sequence. Several approaches have been proposed to detect changes in signals or systems. They include likelihood ratio based approaches such as the Generalized Likelihood Ratio (GLR) test [16] or the marginal likelihood ratio test [17], both effective whenever an accurate and tractable signal model exists and can be implemented. On-line versions based on statistical filtering have also shown good performance [16], [18] while other model-based approaches performing efficient off-line Bayesian segmentation include [19] and [20]. Other general and ad-hoc model-free methods have been designed to detect changes in signals with typical examples being time-frequency approaches [21] and wavelet approaches [22], [23]. Two different methods have been proposed in this paper. The first is a novel "ad hoc solution" consisting of an adaptive/moving threshold test on the instantaneous values of the obtained residuals. The second is a "particle filtering-based likelihood ratio decision solution".

Experimental validation of the proposed scheme has been performed on a differential drive mobile robot, the ATRV-Jr manufactured by iRobot. The robot model has been derived using the inertial navigation system (INS) [24] and odometer equations [25]. Different faults have been investigated and analyzed for different trajectories and tasks. Experimental tests include both single sensor faults and multiple sensor faults. In all cases, the proposed scheme detected and isolated occurred faults within one sample time. Performed experiments have also contributed to a very thorough understanding of occurring time lags between sensor data recordings and residual and change detector calculation that unless taken into consideration, the sensor FDI system will not function properly; this is an additional contribution of this paper. Further, most published papers studying the FDI problem using model-based techniques and structural analysis do not present experimental validation, while simulation results do not consider the residual evaluation problem. Last, but not least, an additional contribution is that the developed sensor FDI system has been integrated with a background distributed architecture that supports heterogeneous robot systems as detailed in [26].

Section II of this paper describes implementation details of the proposed real-time sensor fault detection and isolation system. Significant experimental results are shown in Section III, while in Section IV concluding remarks end this paper.

II. Real-time implementation of the SFDS

A. Background information

The proposed FDI system has been implemented and tested experimentally on a differential drive ATRV-Jr mobile robot platform shown in Fig. 1. The objective of the sensor FDI system is to detect and isolate occurred fault(s) (and inform the user/ground control station if needed). All experiments have been performed outdoors in an environment with several tall buildings (affecting GPS readings), vegetation and palm trees.

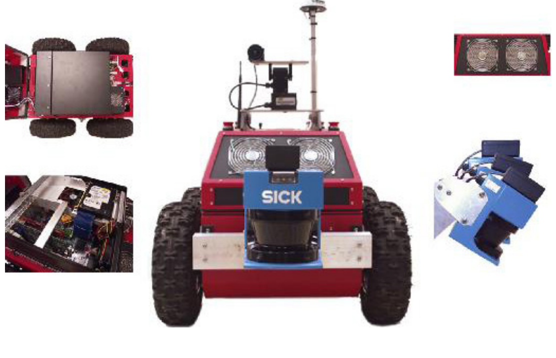

Fig. 1 ATVR-Jr mobile robot.

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, 3.2GHz, 2GB Memory) through a Rocketport multi serial port card. Three sensors are used to evaluate the FDI system: Garmin 16A GPS, internal odometry and Crossbow's IMU 400CC-200, with measured resolutions as shown in TABLE I [27], [28].

TABLE I
SENSOR RESOLUTION

SENSOR	RESOLUTION
IMU 400C-200 gyroscope	< 0.05 °/s
IMU 400C-200 accelerometer	< 1.25 mg
Garmin 16A GPS	< 3 m

GPS, IMU and Odometry data are recorded in real-time as the robot follows the test trajectories shown in Fig. 2 and Fig. 3. Recorded sensor data are then used as input to generate the five residuals on-line and the change detectors following the block diagram steps shown in Fig. 1 of [1].

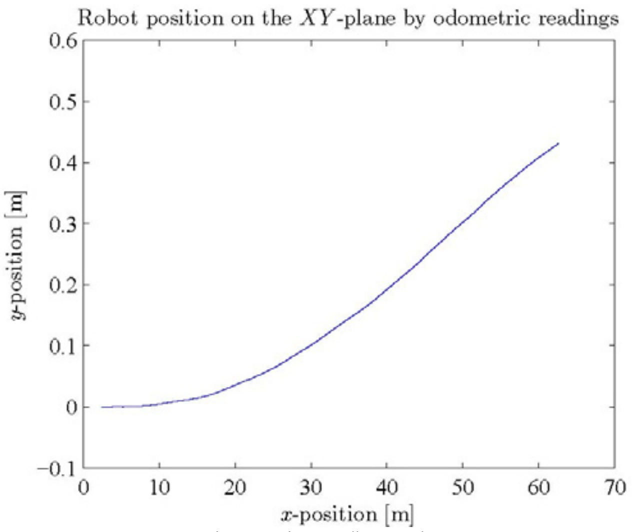

Fig. 2 Trajectory #1 *NE*-plane.

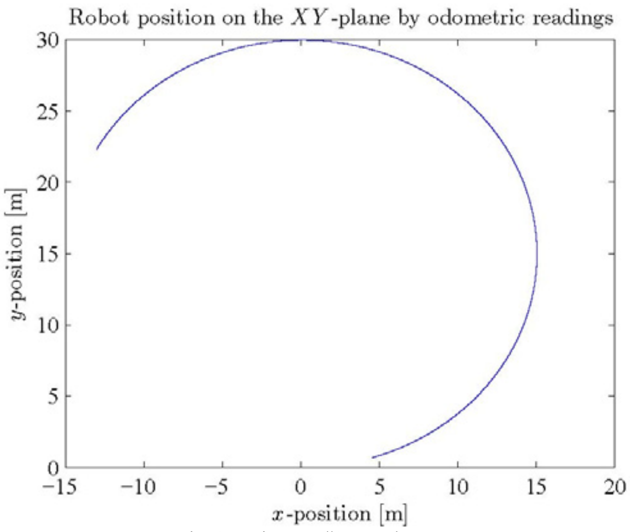

Fig. 3 Trajectory #2 *NE*-plane.

Additive sensor abrupt faults are considered; they are introduced as “user imposed/software generated step faults” added to the recorded sensor data at different times as the robot moves along a trajectory. In essence, after a fixed number of recorded “faultless” sensor data samples, an additive error is introduced into one or more of the sensors.

The magnitude of the step faults is shown in TABLE II; values have been calculated by considering sensor resolution and the signal to noise ratio of each residual.

TABLE II
MAGNITUDE OF THE EXPERIMENTAL STEP FAULTS

PARAMETER	FAULT MAGNITUDE
GPS latitude and longitude	12.75 m
forward acceleration	1.25 mg
yaw angular rate	0.17 °/s
left wheel angular velocity	2.85 °/s
rigth wheel angular velocity	2.85 °/s

An advantage of the overall system in its current configuration is that it may function in “actual” real-time and “pseudo” real-time. The former is obvious, 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 sensor FDI system has been integrated with the already existing Distributed Field Robot Architecture (DFRA) [26], suitable for navigation and control of teams of heterogeneous ground robot vehicles. Since the integration is rather involved and complicated and has resulted in several interesting observations, details are provided next.

B. *Implementation and integration issues*

DFRA extends the SFX managerial architecture [29] and it is implemented in Java/Jini using modular services to implement robot capabilities, including sensors, effectors, and behaviours. 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 [30]. DFRA implements ‘*Sensing Manager*’ module that is responsible for error classification and handling.

The operating system used is Red Hat Linux 9 (RH9), kernel version 2.6.7, a non real time operating system. DFRA uses the standard Java Virtual Machine (JVM) thread scheduling mechanism to handle various service threads within a JVM. The specific low level DFRA 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, perhaps redundant but very useful, 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).

However, since the backbone architecture is in Java/Jini, complete integration, verification and validation of residual and change detector correctness required derivation of two modules developed in Java, called ‘*ResidualCalcOffline*’ and ‘*ResidualCalcOnline*’.

The purpose of the Java based ‘*ResidualCalcOffline*’ module is to receive real-time GPS, IMU and Odometry data and generate residuals and change detectors off-line duplicating in reality what the MATLAB/Simulink module does. This validates in an indirect way that Java based modules and MATLAB/Simulink based modules function identically. The Java based ‘*ResidualCalcOffline*’ module is also being converted to a Java based ‘*ResidualCalcOnline*’ one that generates on-line residuals and change detectors during actual experiments, validating again identical operation in real-time of Java based and MATLAB/Simulink based modules. ‘*ResidualCalcOnline*’ updates the ‘*Sensing Manager*’ with the sensor status. ‘*Sensing Manager*’ passes on the sensor status to a client GUI. 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 failure.

C. *Residual and change detector calculations – observations*

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 GPS, odometry and IMU at a (best effort) sampling rate of 200 msec; this sample time interval is chosen because of the GPS sensor update rate - the slowest of all three sensors; ii) ‘*Residual calculation*’ thread that calculates residuals and change detectors on-line by retrieving data stored by the ‘*Data collection*’ thread. However, close observation of sensor readings reveals that the 200 msec 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; instead, the order is IMU, odometry and GPS, introducing a small time lag between the IMU–odometry and odometry–GPS readings. ii) The Java garbage collector obstructs DFRA 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 shown in Fig. 4, while

Fig. 5 illustrates actual varying sample time when the robot follows the arc trajectory shown in Fig. 3. Fig. 6 and Fig. 7 illustrate observed measured time lags between IMU-odometry and odometry-GPS readings when the robot follows the arc trajectory.

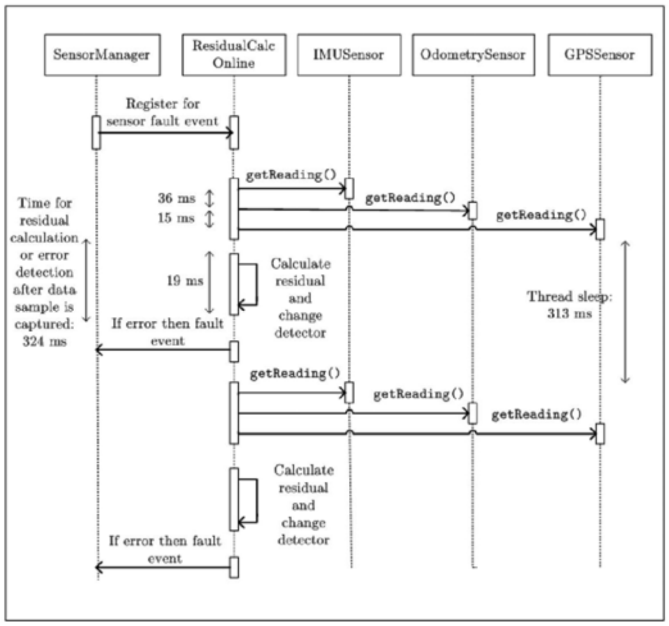
SensorManager
ResidualCalc Online
IMUSensor
OdometrySensor
GPSSensor
Register for sensor fault event
getReading()
getReading()
getReading()
36 ms
15 ms
Time for residual calculation or error detection after data sample is captured: 324 ms
19 ms
Calculate residual and change detector
Thread sleep: 313 ms
If error then fault event
getReading()
getReading()
getReading()
Calculate residual and change detector
If error then fault event

Fig. 4 Sequence diagram for service interaction. The times (in milli seconds) above reflect the observed time delays.

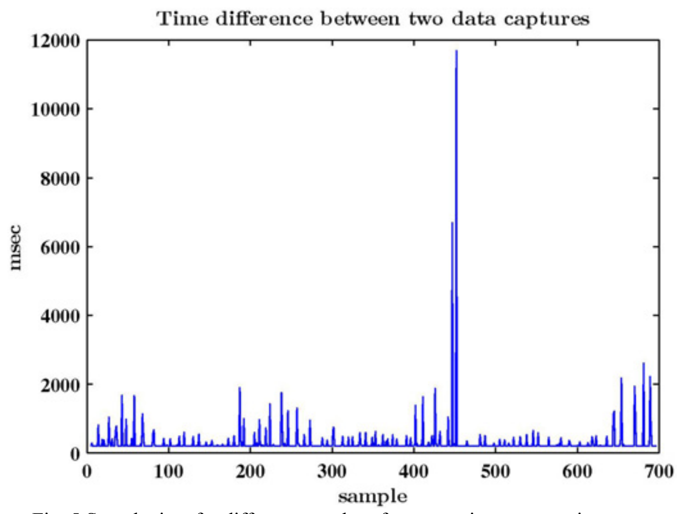

Fig. 5 Sample time for different samples of an arc trajectory experiment.

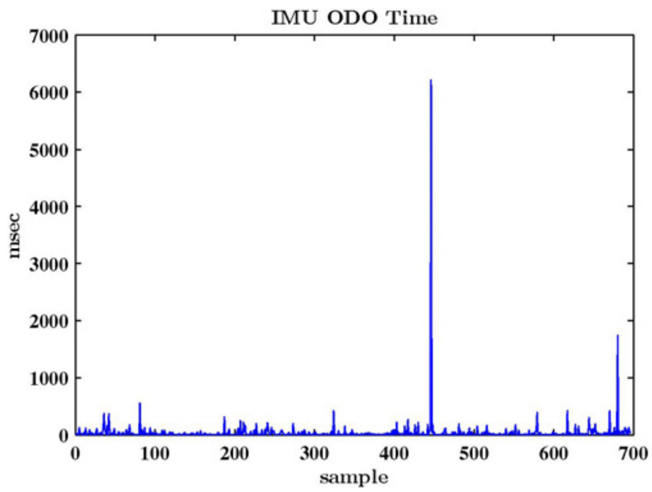

Fig. 6 Time lag between IMU and Odometer readings for different samples of an arc trajectory experiment.

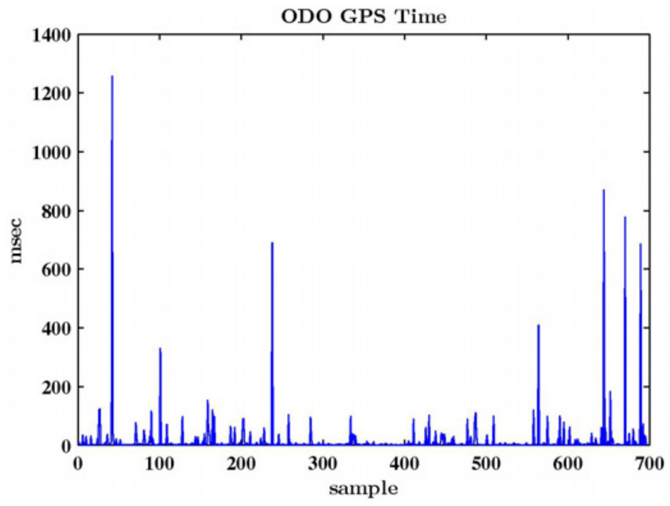

Fig. 7 Time lag between Odometer and GPS readings for different samples of an arc trajectory experiment.

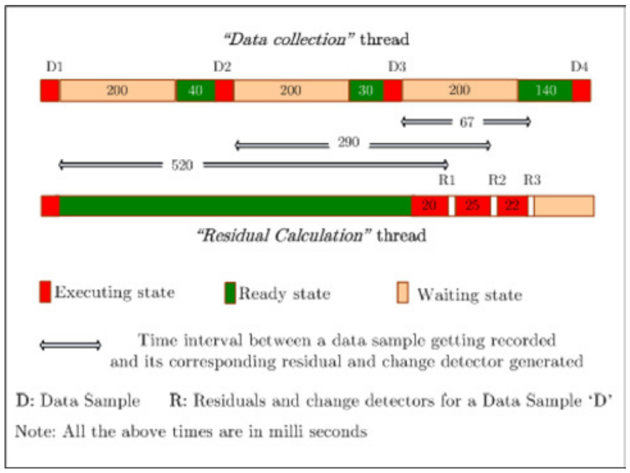
"Data collection" thread
D1
D2
D3
D4
200
40
200
30
200
140
67
290
520
R1
R2
R3
20
25
22
"Residual Calculation" thread
Executing state
Ready state
Waiting state
Time interval between a data sample getting recorded and its corresponding residual and change detector generated
D: Data Sample
R: Residuals and change detectors for a Data Sample 'D'
Note: All the above times are in milli seconds

Fig. 8 The various states of the '*Data collection*' and '*Residual calculation*' threads.

In order to gain further insight to involved delays that may impact the sensor FDI system reliability, Fig. 8 shows the three states of the '*Data collection*' and '*Residual calculation*' threads for three sequential data samples; that is, the Figure displays the time interval between recording a data sample and its corresponding residual and change detector generation.

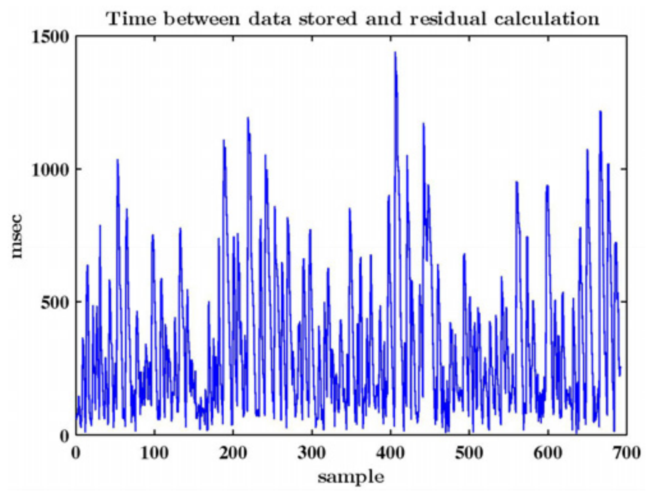

Fig. 9 Time interval between a data sample getting recorded and its corresponding residual and change detector generation for different data samples of an arc trajectory experiment.

For '*Data collection*' the states 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.

For '*Residual collection*' the corresponding states are: *Executing* - the thread calculates residuals and change detectors for a particular data sample (requires <30 msec for each data sample); *Waiting* - the thread is waiting for new data samples to be recorded (this happens after the thread has calculated residuals and change detectors for all data samples collected by the '*Data collection*' thus far); *Ready* - the thread is ready to calculate residuals and change detectors for the new data samples and is waiting for CPU allocation.

Fig. 9 shows the time interval between a recorded data sample and its corresponding residual and change detector for different data samples of the arc trajectory experiment.

III. Experimental results

Sensor data are collected and residuals are generated on-line. For cases with one sensor fault, after a fixed number of data samples, an additive error is introduced into one of the sensors. Experiments consider single and multiple faults per trial following the same principle of introducing additive errors.

The implemented SFDIS 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 is occurred at a time. The situation is slightly different when multiple sensor faults occur simultaneously. In this case, fault isolability cannot be guaranteed in all situations, as it is resumed in the fault signature table, Table III

TABLE III
Effects of the sensor faults on the residuals

	f_{GPS}	f_{ACC}	f_{GYRO}	f_{EL}	f_{ER}
r_1	×	0	0	0	0
r_2	0	×	0	0	0
r_3	0	0	×	0	0
r_4	0	0	×	×	×
r_5	0	×	0	0	×

where f_{GPS}, f_{ACC}, f_{GYRO}, f_{EL} and f_{ER} denote "GPS antenna fault", "accelerometer fault", "gyroscope fault", "left optical encoder fault" and "right optical encoder fault", respectively, while "×" ("0") indicates that the fault in the corresponding column affects (does not affect) the residual of the corresponding row (for further details, see [9]).

Exhaustive and detailed experiments have been conducted in different environment conditions, mobile tasks and trajectories. For brevity, only a few selected results are presented here which are related solely to a straight line trajectory.

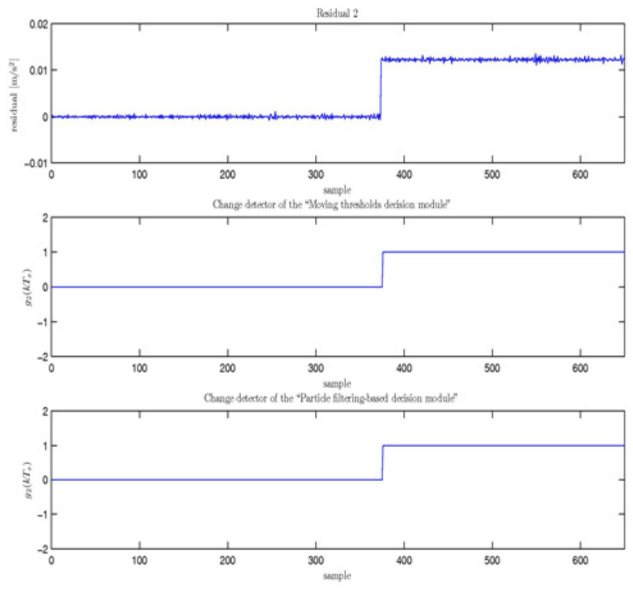

Fig. 10 Comparison between the two proposed residual evaluation modules.

Two different residual evaluation solutions have been considered and experimented in real-time: a novel solution based on adaptive/moving threshold test, and a particle filtering-based likelihood ratio decision solution (as detailed in [1]). Performance of both proposed residual evaluation modules has been analyzed and compared. It has been observed that the two proposed decision modules have shown the same fault detection performances. For example, in Figure 10, the GPS fault has been detected at the same time; the same observation holds for all other experiments. However, the decision module based on the particle filter implementation requires more computational time, therefore, the novel adaptive/moving thresholds decision module may be preferable to use.

Following a series of significant experimental figures which reflect in sequence: "*faultless case*"; "*GPS fault case*"; "*acceloremeter fault case*"; "*GPS fault plus accelerometer fault plus gyroscope fault simultaneously occurring*" and "*accelerometer fault plus gyroscope fault plus right optical encoder fault simultaneously occurring*".

In Fig. 11, the faultless case is presented. It can be seen that all five residuals deviate from zero only due to the measurement noise.

A fault on GPS antenna affects only residual r_1. This can be seen on Table III and it is confirmed by the experimental results reported in Fig. 12. In this experiment, the sensor fault has been generated at 350-*th* time sample, and the fault is detected and isolated at 351-*th*.

Table III shows that accelerometer fault affects both residuals r_2 and r_5 and this is also validated by experimental results shown in Fig. 13. In this case, the accelerometer fault has been generated at 380-*th* time sample and the change detector of the residual issues an alarm after a sample time, which is the minimum possible detection time for the considered system architecture.

Experimental results of multiple sensor faults are presented in Fig. 14, Fig. 15, Fig. 16 and Fig. 17, where Fig. 14 and Fig. 16 shown the residuals, while Fig. 15 and Fig. 17 shown the correspondent decision functions. In the first experiment (see Fig. 14 and Fig. 15), the GPS antenna, the accelerometer and gyroscope fail simultaneously affecting all five residuals. Note that although it is possible to assert that GPS antenna, accelerometer and gyroscope fail for sure (r_1, r_2 and r_3 deviate from zero), nothing it can be said about left/right optical encoder fault. This limitation can be overcame, for instance, adding new sensors so that the redundancy increases. From Fig. 15, it is possible to see how the developed decision module is able to detect the presence of all occurred faults; in fact, g_1, g_2, g_3, g_4 and g_5 issue an alarm only after a time sample.

In the second experiment (see Fig. 16 and Fig. 17), accelerometer, gyroscope and the right optical encoder fail at the same time affecting residuals r_2, r_3, r_4 and r_5, as described in Table III. Fig. 17 shows how the decision function g_2, g_3, g_4 and g_5 issue an alarm, while g_1 does not; in fact, g_1 issue an alarm only when GPS antenna fails, as described in Table III.

IV. Conclusions

The paper presents experimental results of a real-time model-based Sensor Fault Detection Isolation System that has been implemented and tested on a mobile robot platform. Implementation details of the residual generation and evaluation modules have been presented, obtaining a decision function for each residual.

One of the most important aspects and contributions of this paper is the on-line experimentation of the FDI 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.

Experimentation and obtained results suggest that the FDI system is reliable and robust and easily applicable to different mobile robot platforms. An improved experimental architecture is under developing for further research activities.

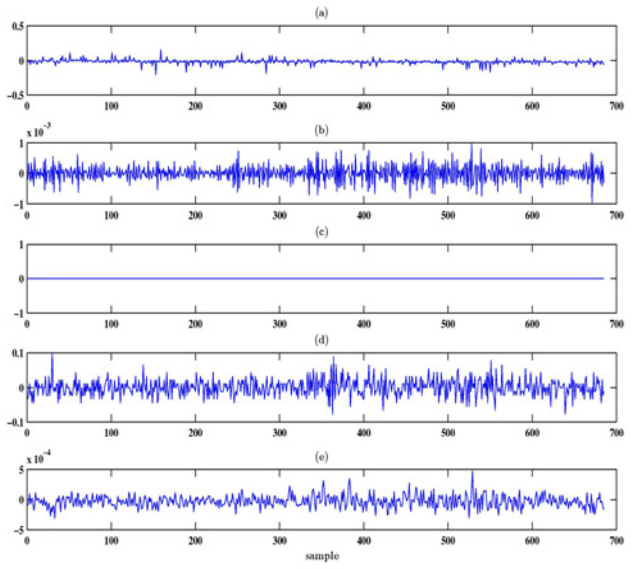

Fig. 11 Faultless case: (a), (b), (c), (d), (e) are residuals r_1, r_2, r_3, r_4 and r_5, respectively.

References

[1] A. Monteriù, P. Asthana, K. Valavanis, and S. Longhi, "Model-based sensor fault detection and isolation system for unmanned ground vehicles: theoretical aspects (part I)", *IEEE International Conference on Robotics and Automation (ICRA'07)*, Rome, Italy, April 2007.

[2] J. Chen and R. J. Patton, *Robust Model-Based Fault Diagnosis for Dynamic Systems*. Boston, MA USA: Kluwer Academic Pub., 1998.

[3] E. Y. Chow and A. S. Willsky, "Issues in the development of a general design algorithm for reliable failure detection," in *Proc. 19-th IEEE Conf. Decis. and Contr.*, Albuquerque, 1980, pp. 1006–1012.

[4] P. M. Frank, "Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy - a survey and some new results," *Automatica*, vol. 26, no. 3, pp. 459–474, 1990.
[5] R. J. Patton and J. Chen, "A review of parity space approaches to fault diagnosis," in *IFAC Safeprocess symposium*, Baden-Baden, 1991.
[6] R. Isermann, "Supervision, fault-detection and fault-diagnosis methods an introduction," *Control Eng. Practice*, vol. 5, pp. 639–652, 1997.
[7] R. J. Patton, P. M. Frank, and R. N. Clark, Eds., *Issues of fault diagnosis for dynamic systems*. New York: Springer-Verlag, May 2000.
[8] R. Izadi-Zamanabadi, "Structural analysis approach to fault diagnosis with application to fixed-wing aircraft motion," in *American Control Conference*, USA, 2002.
[9] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki, *Diagnosis and Fault-Tolerant Control*, ser. Heidelberg. Springer-Verlag, 2003.
[10] A. Monteriù, P. Asthana, K. Valavanis, and S. Longhi, "Experimental validation of a real-time model-based sensor fault detection and isolation system for unmanned ground vehicles", in *Proc. of the 14th Mediterranean Conference on Control Automation (MED 2006*), Ancona, Italy, June 2006.
[11] M. Blanke, V. Cocquempot, R. I. Zamanabadi, and M. Staroswiecki, "Residual generation for the ship benchmark using structural approach," in *Proc. of Int. Conference on CONTROL'98*, Swansea, UK, Sep 1998.
[12] M. Staroswiecki and P. Declerck, "Analytical redundancy in non linear interconnected systems by means of structural analysis," *IFAC Advanced Information Processing in Automatic Control*, 1989.
[13] P. Declerck and M. Staroswiecki, "Characterization of the canonical components of a structural graph for fault detection in large scale industrial plants," *ECC91*, pp. 298–303, 1991.
[14] M. Blanke, H. Niemann, and T. Lorentzen, "Structural analysis – a case study of the Rømer satellite," in *Proc. of IFAC Safeprocess 2003*, Washington, DC, USA, 2003.
[15] A. Monteriù, "Fault-tolerant methods for sensor fusion," Master's thesis, Universit`a Politecnica delle Marche, Ancona, Italy—Technical University of Denmark, Kongens Lyngby, Denmark, 2003.
[16] M. Basseville and I. V. Nikiforov, *Detection of Abrupt Changes: Theory and Application*. Englewood Cliffs, N.J.: Prentice-Hall, Inc., Apr 1993.
[17] F. Gustafsson, "The marginalized likelihood ratio test for detecting abrupt changes," *IEEE Trans. on Automatic Control*, vol. 41, pp. 66–78, 1996.
[18] V. Kadirkamanathan, P. Li, M. Jaward, and S. Fabri, "Particle filteringbased fault detection in non-linear stochastic systems," *International Journal of Systems Science*, vol. 33, no. 4, pp. 259–265, March 2002.
[19] E. Punskaya, C. Andrieu, A. Doucet, and W. J. Fitzgerald, "Bayesian curve fitting with applications to signal segmentation," *IEEE trans. on Signal Processing*, vol. 50, no. 3, pp. 747–758, 2002.
[20] J. Y. Tourneret, M. Doisy, and M. Lavielle, "Bayesian retrospective detection of multiple changepoints corrupted by multiplicative noise. application to sar image edge detection," *IEEE trans. on Signal Processing*, vol. 83, no. 9, pp. 1871–1887, September 2003.
[21] H. Laurent and C. Doncarli, "Stationarity index for abrupt changes detection in the time-frequency plane," *IEEE Signal Processing Letters*, vol. 5, no. 2, pp. 43–45, February 1998.
[22] M. Crouse, R. Nowak, and R. Baraniuk, "Wavelet-based statistical signal processing using hidden markov models," *IEEE trans. On Signal Processing*, vol. 46, no. 4, pp. 886–902, April 1998.
[23] E. Hitti and M. F. Lucas, "Wavelet-packet basis selection for abrupt changes detection in multicomponent signals," *EUSIPCO-98*, 1998.
[24] J. A. Farrell and M. Barth, *The Global Positioning System & Inertial Navigation*. McGraw-Hill, 1998.
[25] J. Borenstein, H. R. Everett, L. Feng, S. W. Lee, and R. H. Byrne, "Where am I? sensors and methods for mobile robot positioning," The University of Michigan, Tech. Rep., 1996.
[26] M. Long, "Creating a distributed field robot architecture for multiple robots," Master's thesis, University of South Florida, November 2004.
[27] [Online]. Available: http://www.xbow.com/index.aspx
[28] [Online]. Available: http://www.garmin.com/manuals/GPS17N GPS16 17NSeriesTechnicalSpecification.pdf
[29] R. Murphy, *Intro to AI Robotics*. MIT Press, 2000.
[30] M. Long, A. Gage, R. Murphy, and K. Valavanis, "Application of the distributed field robot architecture to a simulated demining task," *IEEE International Conference on Robotics and Automation (ICRA)*, 2005.

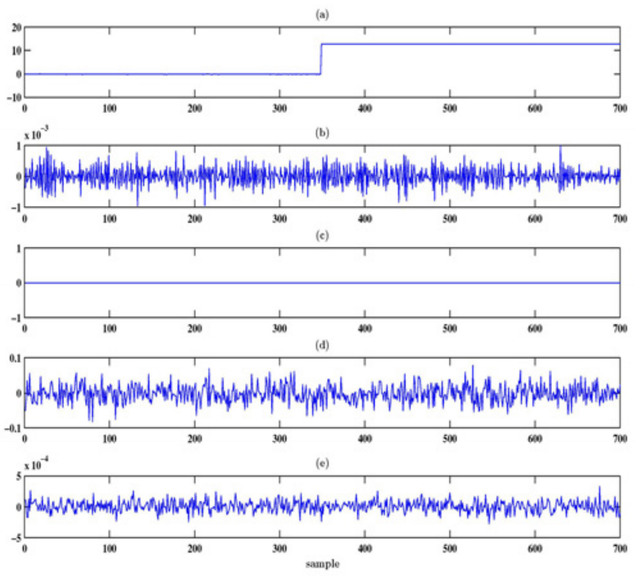

Fig. 12 GPS fault: (a), (b), (c), (d), (e) are residuals r_1, r_2, r_3, r_4 and r_5, respectively.

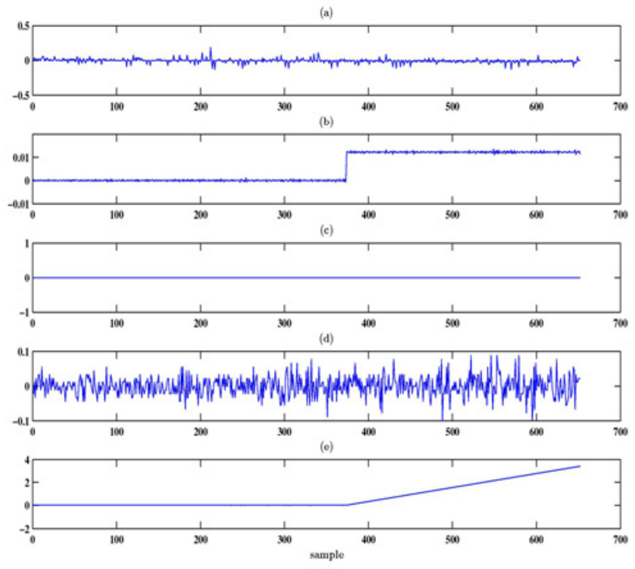

Fig. 13 Accelerometer fault: (a), (b), (c), (d), (e) are residuals r_1, r_2, r_3, r_4 and r_5, respectively.

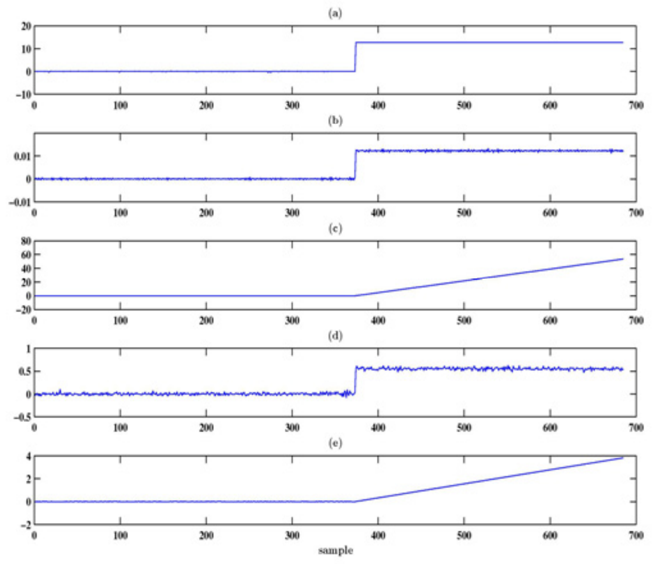

Fig. 14 GPS plus ACC plus GYRO fault: (a), (b), (c), (d), (e) are residuals r_1, r_2, r_3, r_4 and r_5, respectively.

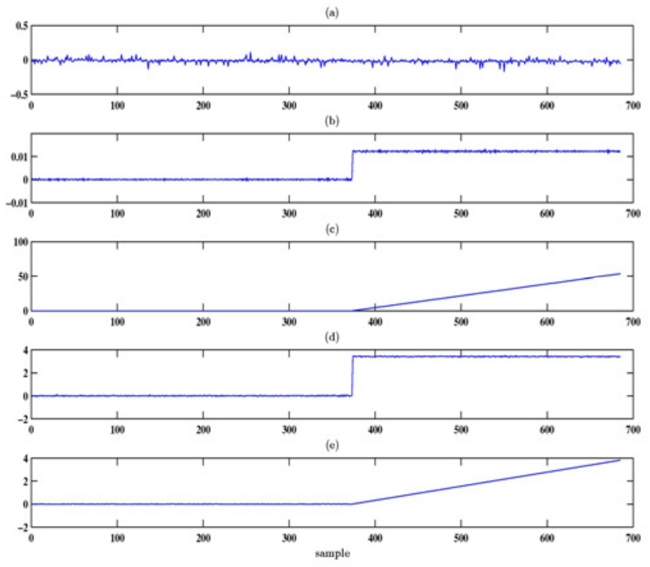

Fig. 16 ACC plus GYRO plus right encoder fault: (a), (b), (c), (d), (e) are residuals r_1, r_2, r_3, r_4 and r_5, respectively.

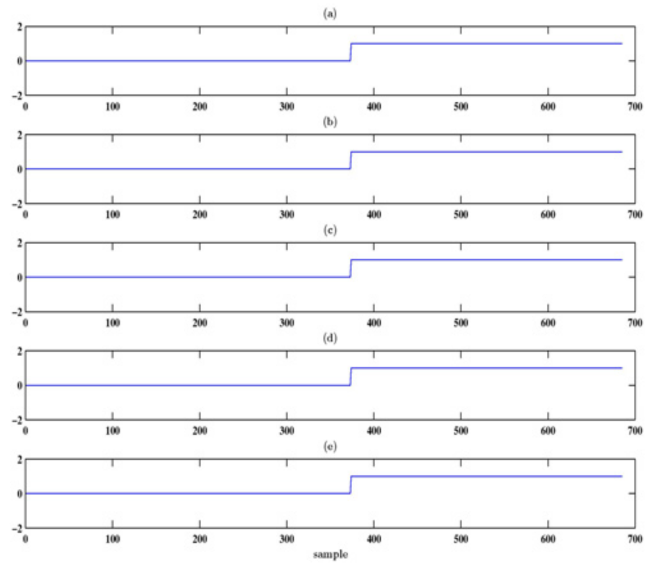

Fig. 15 GPS plus ACC plus GYRO fault: (a), (b), (c), (d), (e) are decision functions g_1, g_2, g_3, g_4 and g_5, respectively.

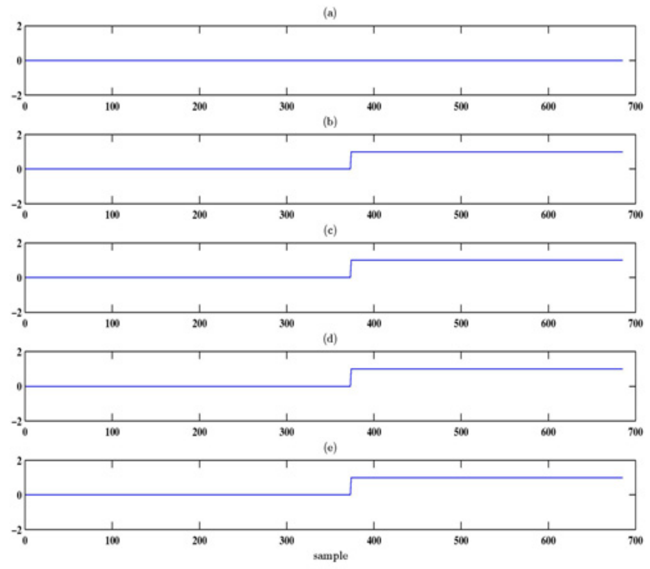

Fig. 17 ACC plus GYRO plus right encoder fault: (a), (b), (c), (d), (e) are decision functions g_1, g_2, g_3, g_4 and g_5, respectively.