Layered Architecture for Fault Detection and Isolation in Cooperative Mobile Robots

Rodrigo A. Carrasco and Aldo Cipriano

Abstract— **This work presents an architecture that can help to increase the reliability in groups of cooperative mobile robots by taking advantage of analytical and sensor redundancy. First, the design of the architecture is portrayed and the faults to be detected are described. The different layers of the system are then explained and analyzed, using simulations to test their capabilities and limitations. Finally, the architecture is implemented on a group of small mobile robots to validate the results of the simulations.**

I. Introduction

THE use of robotic systems in environments hostile to human beings or in dangerous tasks, such as land mines extraction and rescue operations, has increased over the last years, making the work safer for human operators. But the advantages of operating with robotic systems are cut back when faults are taken into account, as they disable the robot in some of its functions, or in worst cases, they make the robot unable to work at all. Recent studies show that the mean time between failures is less than 20 hours for field robots [1], after which they require some sort of repair, consuming time and resources. This implies that an increment in reliability is needed, increasing the mean time between failures or reducing the repairing time.

Two main methods have been used to increase the reliability of robots. One method is to construct more robust mobile robots, which can be done either by adding mechanical or sensory redundancy to the robot. The other method to increase the reliability is to add FDI systems, which identify present problems and thus reduce the time and effort needed for repairs [2]. As these systems are also used in most fault tolerant control systems, as in [3], this work is centred in the design and development of a FDI architecture for cooperative mobile robots, which can obtain information regarding the state of each robot for future control decisions or repairs.

Several FDI methods have been developed for mobile robots [4]. The core of the FDI algorithm used in this paper is presented in [5], where only two possible faults are analyzed: a reduction in the radius of one tire and a periodic bump. In [6], the concept is further developed using a bank of Kalman filters to determine faults on sensors and actuators of a four wheeled robot. Another FDI method is presented in [7], where a bank of Kalman filters is combined with a Markov model representation to identify the faults through probability calculations. Other approaches to FDI in mobile robots include the use of more advanced filtering techniques to identify the faults, which incorporate the nonlinear robot dynamics, as seen in [8].

Also aiming towards an improvement in the reliability, several researchers have proposed a multiple robot approach. In [9], the authors explain how the redundancy present in cooperative mobile robots can be used to increase the robustness of the group and thus improving the efficiency. In the same line of study, the ALLIANCE architecture presented in [10] shows a simple FDI system for cooperative robots based on behavioural programming. Regretfully, with exception of [11], which describes a simple FDI method for cooperative manipulators, none of works mentioned above contain an analysis on the FDI limitations, nor of the efficiency of the isolation (number of false positives or wrongly isolated faults), so no reliability comparisons can be made between them.

This paper presents the basis for a FDI framework for mobile robots, defining an ordered structure over which future FDI methodologies can be developed and implemented. Taking advantage of the different benefits that single and multiple robots' FDI mechanisms show, this work shows a layered architecture for FDI on cooperative robots, where the different layers can be implemented on the robot system depending on the capabilities and resources present. The idea behind this architecture is to combine methods behind single and cooperative robot FDI systems to achieve an architecture capable of detecting a wide range of faults.

The multiple layer approach allows to take advantage of the different information, control, and redundancy levels that exist within the control structure, designing each layer of the FDI architecture according to the level of information available at its corresponding level in the control structure. This permits an efficient use of the available information.

Multiple layers have been used by other authors to achieve FDI in different classes of robotic systems, adapting the FDI system depending on the redundancy that exists, [12]-[13], but the idea of having a cooperative layer within the architecture has not been implemented yet.

Fig. 1 illustrates the framework used, describing the control structure for each robot R_i, according to the level of control, and the interaction with the different layers of the FDI system.

The control structure has five main layers. First, the

This work was supported by FONDECYT project no 1020741, "Fault Detection and Identification in Nonlinear Time Variant Systems".

R. A. Carrasco and A. Cipriano are with the Electrical Engineering Department, Pontificia Universidad Católica de Chile, Santiago, Chile (e-mail: {rax; aciprian}@ing.puc.cl).

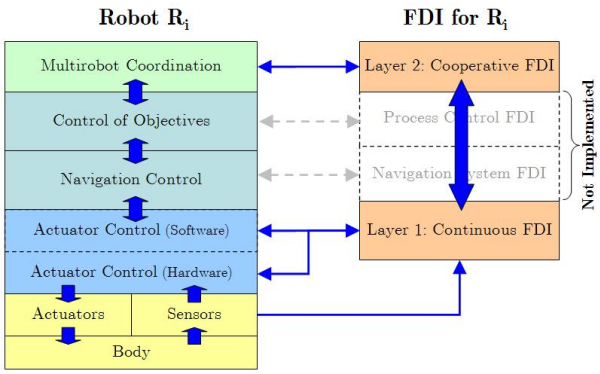

Fig. 1. Control and FDI Structures for robot R_i.

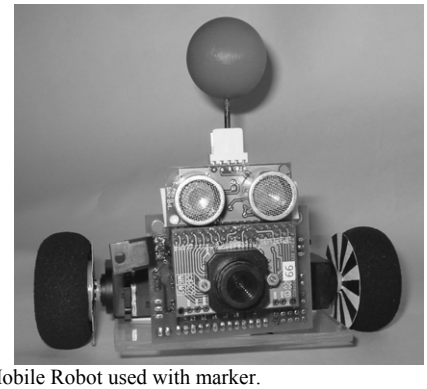

Fig. 2. Mobile Robot used with marker.

physical layer consists of the body, sensors, and actuators needed. The Actuator Control Layer controls the robot's hardware in order to follow a determined trajectory. Next, the Navigation Control is dedicated to design the trajectories needed to achieve the different objectives. The highest control layer in a single robot is the Control of Objectives Layer, which designates the tasks that must be done and where the robot must go in order to do them. Finally, in cooperative robots another layer is added: the Multirobot Coordination Layer. This layer can be either centralized or distributed among the robots, and is the one that designates the objectives of each robot, to achieve their common goal.

This work is divided into five sections. First, section 2 presents a description of the robot system over which the FDI architecture is implemented. Next, section 3 describes the first layer of the architecture, presenting the method used and an analysis of the fault detection capabilities and limitations. Section 4 continues with the description of the second layer of the architecture, indicating how the cooperative robots approach is used. Section 5 then describes the interaction between both layers. Finally, section 6 shows the experimental results of each layer.

II. System Description

The FDI architecture's design is based on simulations, for an analysis of its capabilities, being then implemented on a group of small mobile robots, for validation.

The simulations were done in Matlab, using a the same mathematical model described in [14], which has the kinematic and dynamic equations for each robot. The sensors readings are also simulated in the model by adding noise to the measurements, using a 10% of the maximum value of the measurement as standard deviation.

A group of homogeneous small mobile robots, constructed at our university, is used to test each layer of the architecture and validate the data obtained through simulations. The group is composed by three robots, as the one showed in Fig. 2. Each robot moves using two independent actuated wheels, enabling differential steering. They have two low cost microcontrollers for programming and control purposes, and are equipped with optical encoders on both wheels to achieve relative localization. The robots are also equipped with a digital compass, to measure the heading angle. Each robot has a frontal sonar and a low resolution CMOS camera, for navigation purposes. For this work, the camera is only used to recognize other robots, which is done by identifying the red marker that each one has on top.

Faults can be divided into two groups: those that can be continuously monitored on a single robot, and those that can be detected through cooperation between them. Although some faults can actually be detected through both methods, they are grouped were it is easier to detect them.

For the first layer of the FDI architecture, seven different faults on sensors and actuators are considered: 1-2: slippage of one of the wheels, 3-4: one of the wheels gets stuck, 5: both wheels get stuck, and 6-7: one of the encoders gets stuck (i.e.: the velocity of that wheel is read as zero). The cooperative layer isolates faults on sensors that are redundant in the robot team. This layer is designed to detect four different faults: 1: additive fault on the sonar, 2: the sonar gives a constant value, 3: additive fault on the compass, 4: the compass gives a constant value.

III. Continuous FDI Layer

A. Method Description

The use of multiple models has shown to be a good tool for continuous monitoring of faults in mobile robots. As all the faults this layer must detect can be modelled within a Kalman filter, a bank of eight Kalman filters is used: one for modelling normal operation (M_0), and seven for modelling the faults (M_1-M_7). The basic structure of each model M_i is as follows:

$$M_i \begin{cases} X_{k+1,i} = A_i X_{k,i} + B_i U_k + w_{k,i} \\ Z_{k+1,i} = C_i X_{k+1,i} + v_{k,i} \end{cases} ; i = 0..7 \qquad (1)$$

In (1), $X_{k,i}$ is the state vector for the robot at time k using model i, whereas $Z_{k,i}$ is the measurement. The matrices A_i, B_i, and C_i are the state equation matrices for model i, and U_k is the control input at time k. The process and measurements

additive white noise are represented by w_k and v_k respectively.

To obtain an optimal estimation of the state and measurement vectors of each model M_i, a Kalman filter is applied. Using these estimations, faults are detected by calculating the probability of hypothesis H_i, which states that model M_i represents the actual operation mode of the robot. The conditional probability that hypothesis H_i is true at time $k+1$, is given by the following expression, according to [15]:

$$P_{k+1}(H_i) = \frac{f\left(Y_{k+1} / M_i, \left[Y_0^T .. Y_k^T\right]\right) P_k(H_i)}{\sum_{j=0}^{7} f\left(Y_{k+1} / M_j, \left[Y_0^T .. Y_k^T\right]\right) P_k(H_j)} \quad (2)$$

In (2), *f(.)* is the conditional probability density function of measurement Y_{k+1}, conditioned on the model M_i and the previous measurements, which is given by:

$$f\left(Y_{k+1} / M_i, \left[Y_0^T .. Y_k^T\right]\right) = \beta_{k+1,i} e^{-\frac{1}{2} D_{k+1,i}} \quad (3)$$

With $D_{k+1,i}$ (the Mahalanobis distance at time $k+1$) and $\beta_{k+1,i}$ being defined by:

$$D_{k+1,i} = r_{k+1,i}^T S_{k+1,i}^{-1} r_{k+1,i} \quad (4)$$

$$\beta_{k+1,i} = \left((2\pi)^m \left|S_{k+1,i}\right|\right)^{-\frac{1}{2}} \quad (5)$$

The parameter m, used in (5), is equal to the number of elements in the measurement vector, and $S_{k+1,i}$ is the residual covariance matrix at time $k+1$ for model M_i. The residual $r_{k+1,i}$ is obtained as the difference between the measurements and the estimation of the measurement vector given by model M_i:

$$r_{k+1,i} = Y_{k+1} - Z_{k+1,i} \quad (6)$$

It is important to notice in (2) that if the probability of a certain hypothesis reaches 0, it can not return to another value. To eliminate this problem, the minimum probability is artificially set to 0.0001.

To reduce the computational requirements of the Kalman filters, only three variables are taken into account for the state and measurement vectors: the rotation speed of the robot, and the speed of each wheel.

$$Y_k = \begin{bmatrix} \dot{\varphi}_k & \omega_{1,k} & \omega_{2,k} \end{bmatrix}^T \quad (7)$$

Each Kalman filter uses a simple kinematic relation, which is modified according to the operation mode:

$$\dot{\varphi}_k = \frac{\lambda r_1 \omega_{1,k} - \mu r_2 \omega_{2,k}}{2b} \quad (8)$$

This equation relates the speed of each wheel, with the rotation speed of the robot. In (8), r_1 and r_2 are the radii of the right and left wheels respectively and $2b$ is the axle length. λ and μ are parameters used to represent the different faults: for M_0 λ=μ=1; for M_1 λ=0.4 μ=1; for M_2 λ=1 μ=0.4; for M_3 λ=0 μ=1; for M_4 λ=1 μ=0; for M_5 λ=μ=0; and for $M_{6\text{-}7}$ λ=μ=1 as the fault affects only the measurements and not the process. The value 0.4 used in the model for faults 1 and 2 is determined empirically, to eliminate false alarms due to the small slippage that mobile robots always have.

B. Fault Detection and Isolation

Once the probability for each hypothesis is calculated, FDI is done by using thresholds. First, a fault is detected when the probability of H_0 is smaller than the threshold P_{DT}. This value is a free parameter that allows tuning, affecting the response time of the detection. If the value is too high, the detection is fast (less than 0.2 [s]), but the number of false alarms is important, whereas if the value is low, the detection takes longer, but false alarms are reduced. Due to the effect of noise, the probability of H_0 can sometimes be lower than P_{DT} for some time intervals, creating a false alarm. To reduce this, the detection is activated if the probability of H_0 is smaller than P_{DT}=0.01 for three consecutive time intervals, reducing false alarms, without increasing the detection time too much.

After fault detection is done, isolation is achieved by detecting which probability surpasses a threshold P_{IS}. If the probability of a fault related hypothesis is above P_{IS}=0.99, it is assumed that that fault is present. These values for P_{DT} and P_{IS} were empirically determined through the simulations done, reducing wrongly isolated faults.

C. Simulation Results

Using the mathematical model of a single robot, the continuous FDI layer is tested through several simulations. The system is simulated 1000 times with random chosen operation modes (normal and faulty ones), using a sampling time of 0.1 [s]. These simulations allow a statistical analysis of the performance of the layer. Four different criteria are used: the amount of false alarms, the confusion matrix, the fault detection time, and the fault isolation time.

False alarms indicate the number of times that faults are detected when nothing is wrong. The simulations show that no false alarms appear thanks to the detection criteria used.

The confusion matrix, C_{m1} (9), shows the relation between the faults that appear on the robot and the faults isolated by the FDI layer. Each element c_{ij} of C_{m1} represents the percentage of times the operating mode M_i is isolated as M_j, showing the isolation effectiveness.

$$C_{m1} = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 98.1 & 0 & 0 & 1.9 & 0 \\ 0 & 0 & 0 & 0 & 97.6 & 0 & 0 & 2.4 \\ 0 & 0 & 0 & 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 \end{bmatrix} \quad (9)$$

The results show that only 0.4% of the total simulated operating modes are wrongly identified. The wrong isolation only affects faults 3-4 (stuck wheels), which are isolated as faults 6-7 (encoder faults) respectively, as the effect of both faults is similar during the first time intervals. This confusion can be eliminated by rising the threshold used for isolation, but that would affect the response time.

Table I shows the average detection and isolation times. Due to the method used, the isolation takes slightly longer than the detection in some of the cases. It can be observed that both detection and isolation require only a small amount of time intervals.

D. Interaction with the Control Structure

The interaction between the FDI architecture and the control structure can not only be used to inform the control system that a fault is present. It can also be used to increase the isolation capabilities, obtaining more information about the present state of the robot.

When the robot collides with an obstacle, the wheels can either slip or get stuck depending on the friction coefficient at that moment. In both cases, the FDI system will give a false alarm, as no real fault is present. If no further actions are taken, the robot can be considered disabled even though it can still work.

To eliminate this problem a control routine is added, which allows to differentiate a real fault from a collision. Whenever a slippage or stuck condition is detected, instead of activating a fault detection flag, the FDI system asks for a change in the rotation direction of the affected wheel. If the probability of H_0 returns to be high again, it means that the robot had collided with an undetected object and not that a fault has occurred. Alternatively, if the probability of the faulty condition continues to be high, a real fault is present. This simple algorithm creates a "virtual bumper sensor" that improves the fault isolation capabilities of the layer.

The biggest limitation this first layer has, is that it must be possible to model the effects of the faults within a Kalman filter. If this is not possible, other isolation method must be used, as there will be no residual covariance matrix available for the probability calculation.

TABLE I
AVERAGE DETECTION AND ISOLATION TIMES FOR LAYER 1

Fault	1	2	3	4	5	6	7
Detection [s]	1.1	1.3	0.7	0.6	0.7	0.3	0.3
Isolation [s]	1.1	1.3	0.8	0.8	0.7	0.3	0.3

IV. FDI ON COOPERATIVE ROBOTS

The idea behind this layer, is to take advantage of the sensor redundancy that exists within a cooperative robot group, which can be done by implementing simple routines.

A. Redundant Sensor Fault Detection and Isolation

Fault detection on redundant sensors can be achieved if at least two independent measurements can be made. If the difference between the readings of two sensors is above a threshold D_{TH}, a fault is detected although there is no enough information to isolate the fault. When there are more than two sensors available, the faulty sensor can be isolated from within the group by detecting which has the biggest difference, as indicated in [11].

Because the robots might not always be close together, the Cooperative FDI layer works only when two or more robots meet, testing the different redundant sensors available. If a fault is detected and only two robots are present, it is assumed that both robots have faulty sensors, till a new robot is found. To identify between additive and stuck type faults, the magnitude of both measurements is stored by each robot. If the difference is similar in two independent tests, it is assumed to be an additive fault, being the amount of the fault this difference, and thus achieving fault diagnosis.

As there is noise in the measurement made by the sensors, the threshold must be optimized to reduce the occurrence of false alarms. The problem is that sometimes the threshold needed is higher that the accepted fault tolerance, and reducing the threshold will result in a useless system with a huge amount false alarms. In this case, the use of a multiple measurements test can help to resolve the problem. It was observed experimentally that the sensors used in the robots present an almost additive white noise, with known standard deviation σ_i. Considering this, the difference between the measurements of two sensors, m_i and m_j, has the following error probability distribution:

$$m_i - m_j \sim \mathrm{N}\left(0, \sigma_d^2\right); \text{ with } \sigma_d^2 = \sigma_\mathrm{i}^2 + \sigma_\mathrm{j}^2 \quad (10)$$

If n measurements are done, the standard deviation of the average difference is reduced to σ_n, where:

$$\sigma_n^2 = \tfrac{1}{n}\sigma_d^2 \quad (11)$$

Depending on the desired threshold D_{TH}, the optimal number of measurements needed in order to have a false alarm probability of $1\text{-}P_n$, is determined by (12), allowing the user to set the sensitivity of the system:

$$n = \frac{\sigma_d^2}{\sigma_n^2}; \; with \; \sigma_d^2 = \frac{D_{TH}}{D_P} \quad (12)$$

Where D_P is obtained from the standard normal distribution table such that:

$$\frac{1}{\sqrt{2\pi}} \int_{x=0}^{D_P} e^{-\frac{x^2}{2}} dx = \frac{P_n}{2} \quad (13)$$

If the robot can do only a limited number of measurements, n_{max}, using (12) and (13), the optimal threshold D_{TH} must be increased to keep the same false alarm probability.

$$D_{TH} = \frac{\sigma_d D_P}{\sqrt{n_{\max}}} \quad (14)$$

Every time two robots in the group meet, the sonar and magnetic compass are used to determine the distance and direction of each other. The number of measurements needed is calculated previously using (12), and the average measurement is transmitted between the robots in order to detect the fault. For the sonar, if the difference between both readings is above D_{THs}, a fault is detected, whereas for the magnetic compass, the difference between both readings must be 180 [°], so if the difference is outside the range $180° \pm D_{THc}$, the fault is detected.

B. Simulation Results

Using the same methodology applied to the first layer, the capabilities of the Cooperative Layer are tested. The FDI layer is simulated 1000 times with randomly chosen faults, and on each simulation it is assumed that the robots can detect each other whenever a sensor check is made. For every sensor, the probability P_n is set to 99.99%. The standard deviation for the error on the compass is set to σ_c=0.15[°] and for the sonar is σ_s=0.02 [m]. Given that D_{THc}=0.5[°] and D_{THs}=0.05[m], then the needed values for n are 2 for the compass and 3 for the sonar, to achieve the desired P_n. For this layer only two criteria are used to test the performance: the number of false alarms and the confusion matrix. Due to the fact that P_n is 99.99%, no false alarms appear during the simulations of this FDI layer. On the other hand, the confusion matrix obtained is perfectly diagonal as no confusion can be made in the isolation of the faults. Once a fault is detected, simulations show that it is correctly isolated and the amount of the fault (for additive faults) is determined with 96% of accuracy.

V. Experimental Results

Due to the computational limitations of our robots, both layers are tested offline using data collected from each robot, which is then processed in a computer. The first layer faults are injected in the following way: wheel slippage is done by using the robot over a slippery surface with plastic wheels; the stuck wheel fault is emulated using the robot over thick carpet, with rubber wheels; and finally, the encoder faults are injected by putting a dark piece of paper between the sensor and the encoder wheel.

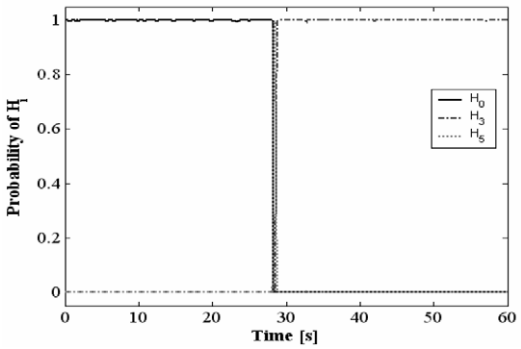

Fig. 3. Experimental result for Fault 3: Right Wheel Gets Stuck.

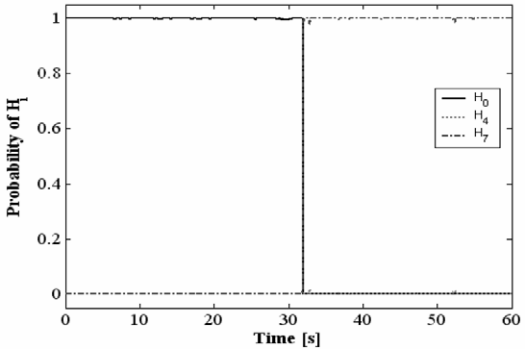

Fig. 4. Experimental result for Fault 7: Left Encoder Fault.

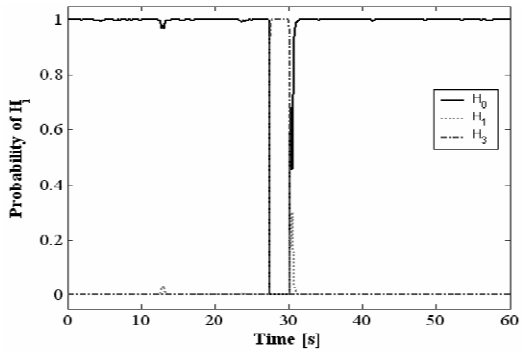

Fig. 5. Experimental result for the "Virtual Bumper Sensor".

Figs. 3 and 4 show the results corresponding to two different faults, injected after about 30 [s]. After the fault is injected, the probability of H_0 reduces, whereas the probability of the corresponding fault increases. It is important to notice that other hypotheses (H_3 in Fig. 3 and H_4 in Fig. 4) also suffer an increase their probability. This effect does not appear in the simulations, and can be attributed to the differences that exist between the parameters used in the model and the ones of the real robot. The system is tested 10 times for every fault, correctly detecting and isolating 100% of the cases.

As this layer is tested offline, the interaction with the control layer must be done artificially. To test the “virtual bumper sensor”, the robot is set on a rough surface and is directed towards a wall, making only the right wheel collide with the wall. After 30 [s] the direction of the wheel is

inverted. Fig. 5 shows the result of the experiment. Following the robot's collision, the probability of fault 3 increases, decreasing when the direction of the wheel is inverted. This indicates that no real fault is present, activating the "virtual bumper sensor".

Before implementing the Cooperative FDI Layer, the sensors are analyzed to determine the standard deviation of the noise, observing that σ_c=0.157[°] and σ_s=1.31 [cm]. To test this layer, robots are set in pairs, detecting each other using the CMOS camera and taking all the measurements needed. Then, another pair of robots is used so the fault can be isolated. Every time, the robot randomly chooses one of the possible faults and injects it to the measurement via software. The values are stored in an external memory for later analysis, and the number of measurements n_c and n_s are calculated using (12). Considering that the tolerated thresholds are defined as D_{THc}=0.5[°] and D_{THs}=3 [cm], with P_n=99.99%. then n_c=2 and n_s=3.

This test is done 30 times in total. The results of the experiments show that no false alarms are activated, and 100% of the faults are correctly isolated. In the case of the additive faults, as the robots stored the amount of the fault injected, it is possible to check the accuracy of the fault diagnosis. In these cases, the amount of the fault is diagnosed with less than 5% of error.

VI. Conclusions and Future Work

Through this work, a layered architecture for fault detection and isolation in cooperative mobile robots is successfully designed and implemented. The reliability of the architecture is measured through simulations, showing excellent results as it is capable of detecting all the simulated faults and isolating correctly 99.6% of them. The architecture is then validated by an off-line implementation, which shows similar results, with no false alarms and 100% accuracy in the isolation of the faults. It is clear that the algorithm used in the Cooperative FDI Layer can also be used to isolate faults on GPS or other types of sensors, without adding too much complexity to the system, which is difficult using other current techniques.

As future work, an on-line implementation of the architecture must be made, which means that a more powerful processor must be added, together with a wireless link that allows information exchange between the robots. Although these first experiments show good results, more experiments in different environments are needed to analyze the robustness of the architecture, specially towards the nonlinearities of the system, uncertainties and perturbations.

About the architecture itself, more layers can be designed and added to take advantage of the information available at other levels in the control structure, and the redundancy existing at the navigation and objective control levels. This could mean for example, monitoring systems that check if the objectives are being achieved, which could help to detect, isolate and even identify new faults (specially those related to robot coordination and trajectory designs).

References

[1] J. Carlson, R. R. Murphy, and A. Nelson, "Follow-up Analysis of Mobile Robot Failures," in *Proc. of the 2004 IEEE International Conference on Robotics & Automation*, New Orleans, 2004, pp. 4987-4994.

[2] R. Isermann and P. Balle, "Trends in the Application of Model Based Fault Detection and Diagnosis of Technical Processes," in *Proc. of the IFAC 13th Triennial World Congress*, San Francisco, California, 1996, vol. 7, pp. 1-12.

[3] Meng Ji, Zhen Zhang, Biswas, G.; Sarkar, N., "Hybrid Fault Adaptive Control of a Wheeled Mobile Robot," *IEEE/ASME Trans. on Mechatronics*, vol. 8, n° 2, pp. 226-233, June 2003.

[4] D. Zhuo-hua; C. Zi-xing; Y. Jin-xia, "Fault Diagnosis and Fault Tolerant Control for Wheeled Mobile Robots under Unknown Environments: A Survey," in *Proc of the IEEE International Conference on Robotics and Automation*, 2005, pp. 3428 – 3433.

[5] S. I. Roumeliotis, G. S. Sukhatme, and G. A. Bekey, "Fault Detection and Identification in a Mobile Robot using Multiple-Model Estimation," in *Proc. of the IEEE International Conference on Robotics and Automation*, Leuven, Belgium, 1998, pp. 2223-2228.

[6] P. Goel, G. Dedeoglu, S. I. Roumeliotis, and G. S. Sukhatme, "Fault Detection and Identification in a Mobile Robot Using Multiple Model Estimation and Neural Network," in *Proc. of the IEEE International Conference on Robotics and Automation*, San Francisco, California, 2000, pp. 2302-2309.

[7] R. Washington, "On-board Real-Time State and Fault Identification for Rovers," in *Proc. of the IEEE International Conference on Robotics and Automation*, San Francisco, California, 2000, pp. 1175-1181.

[8] W. E. Dixon, I. D. Walker, and D. M. Dawson, "Fault Detection for Wheeled Mobile Robots with Parametric Uncertainty," in *Proc. of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Como, Italia, 2001, pp. 1245-1250.

[9] D. G. Michaelson and J. Jiang, "Modeling of Redundancy in Multiple Mobile Robots," in *Proc. of the American Control Conference*, Chicago, 2000, pp. 28-30.

[10] L. Parker, "ALLIANCE: An Architecture for Fault Tolerant Multirobot Cooperation," *IEEE Trans. on Robotics and Automation*, vol. 14, n° 2, pp. 220-240, April 1998.

[11] R. Tinós, M. H. Terra, and M. Bergerman, "Fault Tolerance in Cooperative Manipulators," in *Proc. of the IEEE International Conference on Robotics and Automation*, Washington DC., 2002, pp. 470-475.

[12] M. L. Visinsky, I. D. Walker, and J. R. Cavallaro, "Layered Dynamic Fault Detection and Tolerance for Robots," in *Proc. of the IEEE International Conference on Robotics and Automation*, Atlanta, 1993, vol. 2, pp. 180-187.

[13] M. L. Visinsky, J. R. Cavallaro, and I. D. Walker, "A Dynamic Fault Tolerance Framework for Remote Robots," *IEEE Trans. on Robotics and Automation*, vol. 11, n° 4, pp. 477-490, August 1995.

[14] R. Carrasco and A. Cipriano, "Queen-bee Genetic Optimization of an Heuristic Based Fuzzy Control Scheme for a Mobile Robot," in *Proc. of the First IEEE Latin-American Conference on Robotics and Automation*, Santiago, Chile, November 2003, pp. 61-66.

[15] P. S. Maybeck and P. D. Hanlon, "Multiple-Model Adaptive Estimation Using a Residual Correlation Kalman Filter Bank," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 36, n° 2, pp. 393-406, April 2000.