Fault Tolerant Autonomous Robots Using Mission Performance Guided Resources Allocation

Lotfi Jaïem, Lionel Lapierre, Karen Godary-Dejean, Didier Crestani

2016 3rd Conference on Control and Fault-Tolerant Systems (SysTol)
Barcelona, Spain, Sept. 7-9, 2016

Fault Tolerant Autonomous Robots Using Mission Performance Guided Resources Allocation

L. Jaiem, L. Lapierre, K. Godary-Dejean, D. Crestani*

Abstract— **Real long-term, complex and autonomous mission is still a challenge for robotics. This paper presents an efficient approach enhancing the robot with fault tolerance. It uses performance viewpoints to guide hardware and software resources allocation all along the mission according to faults effects and detection. Simulated and experimental results are proposed and analyzed.**

I. INTRODUCTION

Nowadays long term autonomous and complex missions remain a challenge in robotics. There is no agreed upon the definition of autonomy. We consider in the following that an autonomous robot is able to perform tasks without any human help [1] and is capable to pursue its goals having an active use of its capabilities. This paper does not consider that a robot is able to choose its goals. Long-term autonomy refers to long duration mission. All the system elements which contribute to autonomy have to be concurrently managed, including, as examples, energy supply, computation capabilities and localization accuracy. These missions are usually performed with different tasks. In fact, research works rarely address the question with a holistic approach, preferring focusing on some specific part of the robotic issues (sensing and acting techniques, etc.). Moreover, on the field, autonomy is far from being attained.

But why long term autonomous and complex missions are still a challenge? Long and complex robotic missions in real dynamic environment are difficult to implement. Experimental data and actions differ from the expected ones; internal models are approximated. So along the mission there is a drift between the expected mission progress and the real one. Moreover some unforeseen events can occur like hardware and software faults or obstacles. Finally, autonomous robotic mission must face the problem of energetic autonomy.

So to address long term and complex autonomous missions, the robot must be able to enhance its robustness, while being able to be fault tolerant and managing its energy. In the sequel, to reach these goals the concepts of duration and energy margins are proposed to deal with model approximations. Based on resources redundancy, hardware and software resources allocation guided by objectifs performance constraints is the key factor used to consider fault tolerant, and energetic issues.

This paper is organized as follow. In a first part the relevance of fault tolerance for robotic mission is discussed and some works concerning fault tolerance and resources allocation are mentioned. In a second part, the experimental context used to illustrate the proposed fault tolerant approach for autonomous mission is presented. The next section summarizes the main steps of this approach. Finally, before concluding, simulation and experimental results are exposed and analyzed.

*Authors are members of the Laboratoire d'Informatique Robotique et Microélectronique de Montpellier (LIRMM), UMR 5506, Université de Montpellier, 161 rue Ada, 34 095 Montpellier Cedex 5, France (phone: (+33-0)4-67-41-95-60; e-mail: author@lirmm.fr).

II. FAULT TOLERANCE AND ROBOTIC MISSION

Analyzing many on the field unmanned ground vehicles robotic applications, Carlson et al. demonstrate in [2] that real robotics must permanently face to many failure causes. Physical failures concern effectors, sensors, communication, control system and energy. Human failures relate to interaction and design faults. Depending on the robot type, the technological maturity of the platform and the environment context, these faults can occur frequently. According to the fault severity, the impact on the robot task can be limited or fatal. In [3], R. Parasuraman lists a set of hardware and software failures which may occur any time during the usage of a mobile robot. The importance of energy management is also pointed out. So it seems evident that autonomous robotic systems must be fault tolerant.

In [4], a large state of the art summarizes the different fault tolerant architectures. The fault tolerance principles are enunciated. Firstly a *fault detection* phase detects possible failures. *Fault isolation* locates the faulty component and a *fault identification* identifies the fault and its severity. The robotic system is then reconfigured to maintain acceptable performance in a *fault recovery* phase (if possible). Then, an adaptive robotic control architecture aiming to enhance the fault tolerance of autonomous mobile robot is proposed. Depending on the severity of the detected fault and of the resources redundancy (including the operator) the architecture is reconfigured on the fly. However, the selection of the embedded recovered resources is predefined by the user's empirical experience and does not respond to an objective optimization. Moreover the reconfiguration concerns the current task without taking into account the mission context, global performance constraints or the remaining tasks. [5] proposes a fault tolerant and self-adaptive reconfiguration architecture based on performance estimation/evaluation on task level. However, the global mission constraints are not considered.

ALLIANCE is a fully distributed behavior-based architecture [6] addressing fault tolerance for multi-robot cooperative tasks. In response to failure, the distributed task control reallocates tasks between robots. It is based on motivational

978-1-5090-0658-8/16/$31.00 ©2016 IEEE

behaviors. Experiments are realized with 3 identical wheeled-robots maximizing redundancy for hazardous waste cleanup mission. This architecture demonstrates its ability to deal with robot removing or inefficiency to realize the mission. However internal robot faults are not really managed and even if a task can be implemented using different ways, the energy issue is not considered. An other fault tolerant multi-robot application is addressed in [7]. Due to faults, the system changes and adapts the robotic system according to the available ones. It is obvious that multi-robot system missions brings redundancy and presents a higher fault tolerance and robustness capacities than single robot missions but they are relatively expensive and their implementation is complex.

In [8], Gspandl et al. analyze the problems that can be encountered in autonomous intelligent systems. They identify the faults of hardware (damage) and software (run-time faults) part of the systems. But furthermore problems can come from the interaction between the system and its environment. They distinguish the sensing fault when observation does not reflect the reality, exogenous events (environment change), execution fault when the action is not the expected one and incomplete or wrong knowledge fault.

III. EXPERIMENTAL CONTEXT

Without losing generality, the performance guided resources allocation methodology proposed in section IV will be illustrated on the following robot, mission and performance context.

A. The Robot Platform

The experimental platform is a two wheeled Pioneer 3DX robot supporting 16 ultrasonic sensors and 10 bumpers. To complement its sensing capacity, two laser sensors (URG-04 LX) provide a full 360° proximity scanning, and a camera (Kinect©) is used for image analysis of geo-referenced QR-codes. An embedded watt-meter measures the robot battery energy consumption. Two USB-Relay boards permit switching on/off the lasers and the Kinect. A laptop, having its own battery, supports the robot control architecture and is carried by the mobile platform. It implements scheduled modules which are executed in periodic schemes according to real-time constraints and mission objectives.

B. The Patrolling Mission

The experimental mission is an autonomous patrolling (Fig. 1) of 187 meters to inspect the state (open/close) of two valves (V1-V2). QR-Code markers can be used for localization.

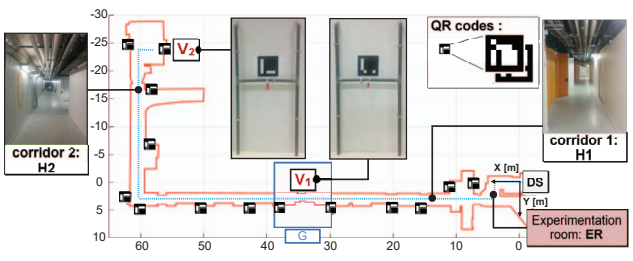


Fig. 1. The Patrolling Mission Description

Different areas are identified: in H1 human can be encountered, but not in H2. G is a glazed area imposing the use of sonars. The mission begins from the docking station (DS). The robot goes to V1 then to V2 and analyzes their states. At the end, the robot goes back to DS.

C. Autonomous Robotic Performance

Many performance indicators have been defined in industrial robotics where the environment is well known, static (without unforeseen elements) and energy can be considered as infinite. The mobile robotic context is quite different since the environment can be partially known (or unknown, for exploration), dynamic (presence of unforeseen and moving objects), and the energy supply is limited. Today there is a lack of accepted metrics for autonomous missions. So we propose to decompose the performance concept into *main frame* and *user's performances*. The *main frame performances* must be always present whatever the considered robot or autonomous mission (**stability**, **safety**, **localization** and **energy**). *User's performances* depend on the user's interest. For example, mission **duration** is a classical viewpoint.

In the following, the study focuses on all these axes. Stability is considered by supposing that a control loop of 10 Hz is sufficient to ensure the control stability of the robot. Then the following performance objectives are considered.

- Safety: the robot moves in a safe way for itself and its environment.
- Energy: the robot must have enough energy to complete its mission (must be less than E_{max}).
- Localization: the robot must be able to locate itself.
- Duration: the mission duration must be less than D_{max}.

IV. THE PROPOSED METHODOLOGY

The methodology consists on two phases :

- Offline performance estimation:
  - building a detailed mission scenario integrating the projection of performance constraints.
  - computing an initial allocation solution (if it exists).
- Online performance evaluation and resources management: monitoring phase consists on:
  - periodical check of events like performance drifts or resources failures.
  - recompute an allocation solution between the available configurations according to the remaining mission.

A. Some Definitions

We consider that a mission can be initially designed as a sequence of n_{obj} **objectives**. Each objective is carried out using a set of n_T robotic **task(s)** (Table I). Each robotic task can be implemented using one of n_{OT} Option Tasks OT allowed by hardware (sensors, actuators) and software (algorithms) resources redundancy. So, an objective can be executed choosing a unique resources configuration for each of its task considered as **alternative implementation AI**. The number of possible AI for an objective is denoted with n_{AI} and expressed in (1).

$$n_{AI} = n_{OT_1} \times \cdots \times n_{OT_{n_T}} \quad (1)$$

TABLE I
TASKS DESCRIPTION

Task Name	Action	OT (algorithms / sensors)	n_{OT}
FM	Forward motion	Path following SMZ [9] (US, 1 LAS, 2 LAS, 2 LAS+US, none) Centering moving (2 LAS, 2 LAS+US)	7
R	Rotation	On place rotation	1
L	Location	Odometer (none), **G**rid **B**ased **M**ethod GBM (2 lasers) and **QR C**ode based **M**ethod QRCM (kinect)	3
IA	Image Analysis	Valve detection algorithm (kinect)	1

However, an objective can be carried out under a set of physical (PhC), legal (LC), environmental (EC) and performance constraints (PC), excluding the use of some sensors or algorithms and limiting or fixing the value of some mission parameters. In Table II we can see that the patrolling mission is initially decomposed into 9 objectives.

The projection of all the constraints on the mission progress allows to divide it (and consequently the objectives) in a sequence of n_{act} ($\geq n_{obj}$) **activities** $A_k^{c_j}$ supporting an invariant set of constraints c_j. An activity is identified by the linear coordinates of its starting and ending points (linear evolution of the mission with respect to the starting point DS), set of AI, velocity limit, etc. This activity sequence is called **Nominal Mission Plan (NMP)**. Even for mission search example, this plan is still needed for the robot come back.

Due to a lack of place, the constraints projection process cannot be detailed but the following constraints have been identified for the studied experiment:

- (PhC) The maximal robot velocity is 0.75 m/s.
- (EC - PC) To ensure safety, sonars must be used in the glazed areas.
- (LC) To ensure safety and harmless in case of collision with dynamics obstacles, possibly human, the impact energy must be less than 4J.
- (PC - LC) To ensure safety and obstacle avoidance of static obstacles, the used avoidance approach (SMZ [9]) imposes that the robot velocity must be less than 0.32 m/s if sonars are used and 0.4 m/s for laser sensors, which offers better precision and sampling frequency.

The spatial definition of these constraints divide the considered mission into 17 activities (Table II).

Velocity limitation by activity V_{max} is expressed with different colors in Table II: blue for 0.32 m/s, green for 0.4 m/s, orange for 0.75 m/s and grey for 0 m/s (static activities). Now the fault tolerant performance guided process can start.

B. *Offline (initial) Performance Estimation*

Once the NMP is built, the question can be summarized as following: How to choose and parameterize an alternative implementation AI for each activity of the mission to satisfy the defined performance objectives?

TABLE II
PATROLLING MISSION DESCRIPTION

x_i - x_{i-1} (m)	0 - 34			37	37	37	37 - 93.5				93.5	93.5	93.5	93.5 - 187			
Objective O_i	DS → V1			↻	⋈	↻	V1 → V2				↻	⋈	↻	V2 → DS			
n_{AI_i}	21			2	1	2	21				2	1	2	21			
Task T_k	FM / L			R L	VD	R L	FM / L				R L	VD	R L	FM / L			
$A_k^{c_j}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
n_{AI_k}	21	9	3	2	1	2	9	21	21	7	2	1	2	21	21	9	21
$V_{max}(m/s)$																	

The main common leverage to the performance axes is the system velocity v. Then, a first question is how to fix the parameter value (velocity v) of a task configuration AI_k for each activity, insuring the respect of the performance objectives defined previously and hence a guarantee a successful execution of the mission. To answer this question we define a Duration Margin which is positive until the mission remains feasible within the maximum duration D_{max}.

The **Duration Margin** (DM) is the difference between the estimated/real mission duration and D_{max}. Maximizing DM, is to choose, for each activity, the highest possible velocity (minimizing the activity duration), while satisfying the safety constraints and optimizing the energy consumption. This energy consumption is estimated using the experimental identification process proposed in [10]. It allows an estimation of the energy cost (robot and laptop batteries) of all the AI_k of an activity k.

Now, for each activity, considering the chosen velocity value, we are able to evaluate the activity duration and its corresponding energy consumption. However, the second problem to consider is the global size of possible alternatives (mission level). This Global Number of local Choice (GNC) is equal to the product of the number of alternatives n_{AI} by activity. For the considered mission, GNC $> 2 \cdot 10^{13}$. This problem can be modeled as knapsack problem.

To address this complex NP-hard problem, we use the algorithm proposed in [11] to quickly found a feasible solution (but not necessary the best one). For each activity, it sorts the energetic consumption of each AI according the energy viewpoint. Then, remarking that global energy composition law (additive) preserves the local ranking, it adjusts a local activity selection using a binary search algorithm to satisfy globally the performance constraint. The most energetic local choice while globally satisfies the energetic objective is selected. This strategy supposes that more energetic an alternative is, most efficient the related robot tasks are. So, if the **Energy Margin** (EM) is defined by the difference between the estimated/real mission energy and E_{max}, the proposed strategy minimizes EM.

Then, this algorithm determines for each activity the hardware and software resources allocation, if a feasible solution exists, and a velocity profile (sequence of v) globally satisfying the defined performance objectives. This planning is called **Resources Allocation Solution** (RAS). Due to the algorithm simplicity and efficiency the RAS computing can

be realized online. The mission can now really start.

C. Online Performance Evaluation and Resources Management

Duration and energy margins are periodically checked along the mission. Different events can lead to the search of a new RAS using the chosen knapsack algorithm on the NMP built from the current position of the robot to the end of the mission. The possible events are:

- Negative energy or duration margins. This event occurs due to different causes. The energetic model accuracy is limited. Moreover, the robot trajectory is necessarily different from the planned one inducing a loss of time and an increase of energy consumption. Finally, obstacles will also lead to a decrease of performance margins. Then, a new RAS is computed adapting resources allocation to restore positive margins.
- A hardware device or a software module becomes faulty. Then the alternatives using the faulty resource are eliminated from the available activity solutions used by the knapsack algorithm. Then the new computed RAS do not involve the faulty element and can be executed.

Obviously, if no new RAS is possible, the mission aborts and the operator is warned.

V. RESULTS

A. Faults and Robotics

Our approach allows enhancing the robustness with regards to different fault classes. Energy and duration margins minimize the direct impact of sensing faults (odometer information are necessarily drifting and generate an increase of traveling path and energy consumption), execution faults (the planned path is never exactly followed), incomplete knowledge (the consumption models are an approximation of the reality) and exogenous events (obstacle avoidance increases the traveling path and energy consumption). So margins act as a stock of duration and energy which can be used to limit the direct impact of interaction faults and gives an immunity degree with regards to them. When the impact is too important (negative margins), a new RAS is computed restoring a positive immunity. A new RAS is also computed when faulty hardware and/or software elements are identified. We must note that this methodology does not require of minimum number of redundant resources but it still a important factor to make the system more robust.

Next paragraphs show some simulated and experimental examples of the proposed fault tolerant approach.

B. Nominal Mission Plan

The NMP is built considering the performance constraints. It consists on a definition of a set of alternatives that can be used for the mission (cf. Table II). The number of alternatives n_{AI_k} reflects a tolerance capacity (resources redundancy) for the considered activity k. The glazed area G impacts the constraints (maximum velocity and eligible AI) from safety viewpoint. The velocity limit becomes 0.32 m/s in this glazed area and alternative implementations AI without sonars are excluded. Otherwise, the velocity limit is 0.4 m/s in H1 (human) and 0.52 m/s in H2 (no human). It is noticed that objectives initially with 21 AI keeps their same set of AI outside G area (V_{max} in blue). n_{AI} for A_2 becomes 9 in G (sonars must be used) and falls to 3 when the Kinect is imposed for accurate localization while approaching V1 (A_3). Hence, the O_1 is decomposed into 3 activities and O_5 into 4 activities and the final objective O_9 is performed with 4 activities since it crosses H2, H1, H1/G and again H1.

C. Simulation Results

The patrolling mission and robot described in section III are considered to evaluate the proposed methodology of fault tolerance using performance guided resources allocation. Energy (robot and laptop) and duration constraints are $E_{robot_{max}}$ = 2.3 Wh, $E_{laptop_{max}}$ = 3 Wh and D_{max} = 580 s. Several obstacle avoidance (OA) are generated along the mission (Fig. 2) and Kinect then $laser_1$ faults (sensor faults SF) occur. Considering these performance constraints and the available hardware and software resources, a first resources allocation solution is computed RAS_0.

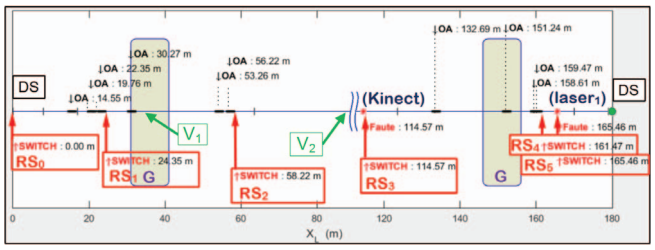


Fig. 2. Mission progress and events

Table III recapitulates the details of the computed solutions. The following shapes indicate the localization algorithm (Table I) and if the following sensor is used or not:

☒/☐: used/unused sensor (none), ●/○ : used/unused sensor (GBM), ▶ / ▷ : used/unused sensor (QRCM).

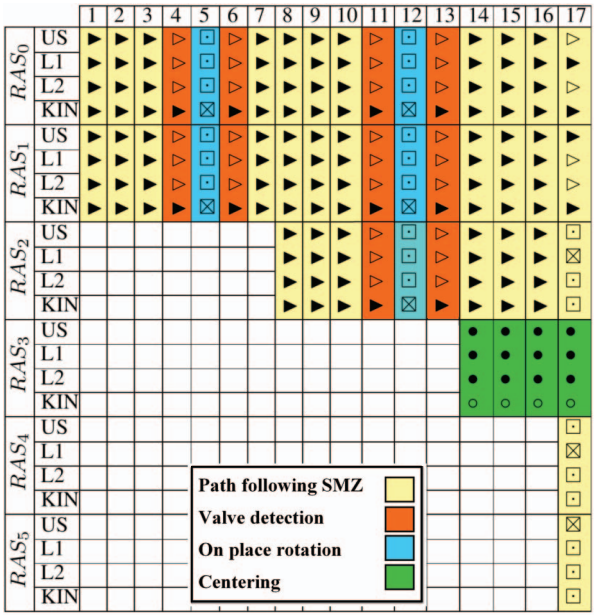
Unexpected obstacles cause a higher energy consumption regarding to the estimated one (Fig. 3 (a) and (b)). These obstacles cause also a loss of time compared to the estimated one (Fig. 3 (c)). Margins decrease and new resources allocation solutions (RAS_1, RAS_2 and RAS_4) are calculated to overcome the environment unforeseen events (OA) and meet performance constraints.

Firstly RAS_1 is computed and the planned moving algorithm for the last mission activity A_{17} is switched from SMZ using $laser_1$ (L1) (RAS_0) to a less consuming one using only sonars (US). So the energy margin is again positive and the mission remains feasible. The second time the energy margin falls negative, RAS_2 is computed: $laser_1$ is activated again and Kinect (KIN) is disconnected for the last activity where only odometers are used for localization.

Other type of events occurs when we randomly generate two sensor fails (Kinect at 114 m and $laser_1$ at 165 m). Since these sensors are used for current and future activities while realizing the mission with RAS_2 and RAS_4, new solutions are calculated to face these events. For the Kinect

TABLE III

RESOURCES ALLOCATION SOLUTIONS FOR SIMULATED MISSION



fail event, the RAS_3 is computed, switching both the moving and localization algorithms to kinect free AI since the Kinect is no longer available. In this case, the energy constraints allow the use of $laser_2$ (previously not used). Then the GBM (Table I) localization option task is activated and the moving task is performed with centering motion. The last

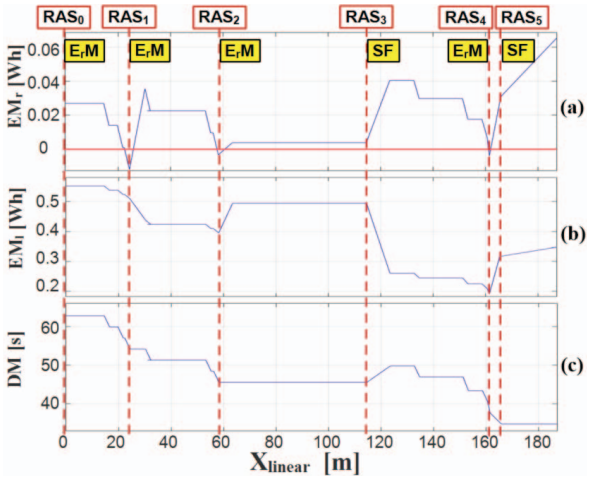


Fig. 3. Margin variations and events for simulated mission

energy margin event occurs at 161 m. So, a new allocation is needed and RAS_4 solution switches off the second $laser_2$ used (with $laser_1$) for localization and centering. The robot runs with odometer localization and path following with obstacle avoidance algorithm based only on $laser_1$ data is used instead of centering algorithm.

The second sensor failure disqualifies the RAS_4 that uses $laser_1$ and the computed RAS_5 activates the sonars. We note an increase of laptop and energy margins since the sonars consumes less than lasers on the battery laptop and the same thing for their corresponding power consumption on the laptop battery. However, the use of sonars imposes a robot velocity decrease for safety reasons (cf. section IV). That explains the fall of duration margin that nevertheless remains positive.

This simulation shows how the robot can be tolerant facing resources failures and environment events. The switch to other alternatives is done while respecting all the given constraints. In the sequel, this methodology is implemented on the robot controller and evaluated on a real experiment while additional type of fault due to incomplete knowledge (model approximation) generates an experimental drift of EM and DM.

D. Experimental Results

A real patrolling mission is performed where the methodology is implemented in the robot controller and additional real constraints are considered like robot drift (additional traveled distance), oscillations (turns), energy estimation error and localization accuracy limits.

The robot has to execute its mission in a safe way under 600 s. The energy should not exceed 2.4 Wh for the robot battery and 2.8 Wh for the laptop.

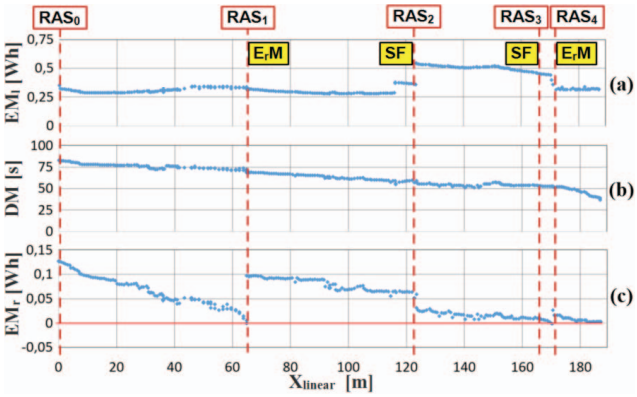


Fig. 4. Margin variations and events for the patrolling mission

Based on the NMP build, the first offline resources allocation RAS_0 is done (NGA $> 2 \cdot 10^{13}$). It is computed with only 681 iterations IT. The details of the computed RAS are recapitulated in Table IV. Continuous margins are initially positives for duration (DM = 82 s) and for both laptop (EM_l = 0.35 Wh) and robot (EM_r = 0.12 Wh) energies (Fig. 4). These margins are based on the nominal execution of the mission with RAS_0.

The mission starts and the energy margin decreases due to obstacle avoidance and velocity variations while leaving the experimental room ER. This noticed difference is due to the gap between estimated energy and duration (theoretical constant velocity and moving in straight line) and the real consumed energy (velocity saturation while turning and over path traveled while oscillating). The estimation fault leads the robot energy margin to be negative at 68 m (Fig. 4 (c)). Then, the robot computes a new resources allocation solution

RAS_1 for the remaining mission activities. Robot energy margin becomes positive again. The mission continues and the robot goes to the second valve at linear coordinate 93.5 m, inspects it and returns to origin (DS).

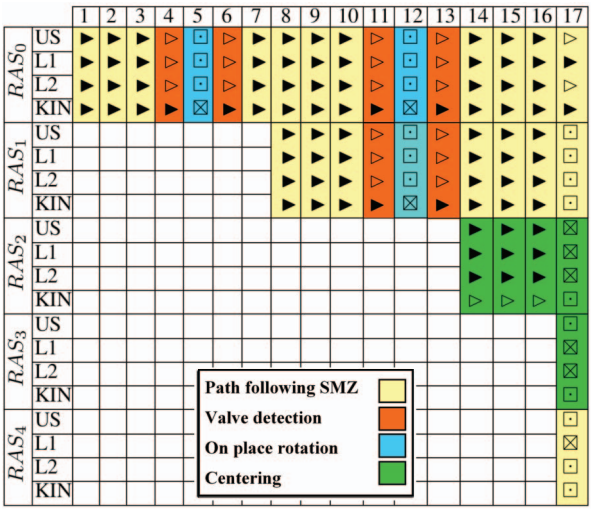
The robot comeback begins in a human free area (H2) but robot drift and localization errors leads the robot relatively to erroneous position and heading. The robot enters into obstacle avoidance with corridor walls until it corrects its localization data based on the used algorithm. At 121 m, a Kinect fault is triggered. The current RAS becomes invalid since the Kinect is used for localization in the current and future activities. The robot computes autonomously RAS_2 by considering the available sensors/algorithms, robot state and performances global constraints. RAS_2 switches the localization algorithm to GBM (using two lasers) for activities A_{14} to A_{16}. The last activity A_{17} is planned to be performed with odometer. Then, the mission can continue.

A second failure (sonars) is triggered at 164 m. Since it is used for A_{17} for centering motion, RAS_3 keeps the same motion algorithm without sonars. If sonars fail before or during passing the glazed area G, the mission will abort from safety viewpoint because this sensor is crucial to detect transparent surfaces and there is no redundant sensor with that ability (laser beams are inefficient).

At the end, robot energy margin becomes negative due to obstacle avoidance and localization error. The last switch occurs by passing from RAS_3 to RAS_4 at 166 m. The localization algorithm is still using odometer but the motion algorithm becomes SMZ path following with $laser_1$ data since the $laser_2$ is switched off to meet, between others, robot energy constraint.

TABLE IV

GENERATED RESOURCES ALLOCATION SOLUTIONS FOR REAL MISSION



The mission succeeded since all margins are positives and the robot consumed less than the required limits. The robot dynamically adapted its configuration to overcome different internal and external events and faults. The difference between estimated and real performances was an origin of faults since the mission has to be performed under different constraints. Hardware and software faults can also induce mission resources reconfiguration. Patrolling mission shows up, in addition to faults tackled for simulated mission (SF and negative margin due to OA), other type of faults like energy and duration prediction due to models approximations and localization errors that lead the robot to enter in obstacle avoidance with static parts of the environment.

VI. CONCLUSION AND FUTURE WORKS

This paper presents an approach to build fault tolerant mission satisfying performance constraints. Different faults are considered (resources availability and efficiency, performance drift due to models approximations, environment unforeseen events like obstacles, etc.). This problem can be modeled as a NP-hard knapsack problem. Duration and energy margins allow dealing with incomplete knowledge faults (model limitations) and exogenous events (obstacle avoidance). Hardware and software faults are considered using an efficient algorithm addressing the knapsack problem and to manage resources allocation. The efficiency of the proposed approach has been demonstrated on simulated and experimental mission examples.

However this work concerns only the recovery aspect of a classical fault tolerance method. It must be afterward completed using fault detection and diagnosis mechanisms. So, the next step of this study is to integrate the proposed recovery approach in the fault tolerant robotic architecture proposed in [4].

REFERENCES

[1] R. Parasuraman, T. Sheridan, and C. Wickens, A model for types and levels of human interaction with automation, IEEE Transactions on Systems, Man, and Cybernetics-Part A: Sytems and Humans, 30(3), pp. 286-297, May 2000.

[2] J. Carlson and R. Murphy, ?How UGVs physically fail in the field, IEEE Transactions on Robotics, vol.21, no.3, pp.423-437, June 2005.

[3] R. Parasuraman, Few common failure cases in mobile robots, arXiv preprint arXiv:1508.03000, 2015.

[4] D. Crestani, K. Godary-Dejean and L. Lapierre, Enhancing fault tolerance of autonomous mobile robots, Robotics and Autonomous Systems, 68, pp. 150-155, 2015.

[5] Y. Cui, J. Lane, R. Voyles, and A. Krishnamoorthy, A new fault tolerance method for field robotics through a self-adaptation architecture, Safety, Security, and Rescue Robotics (SSRR), IEEE International Symposium, October 2014.

[6] L. E. Parker, ALLIANCE: An architecture for fault tolerance multi-robot cooperation, IEEE Transactions on Robotics and Automation, 14 (2), pp. 220-240, 1998.

[7] D. Portugal, and R. P. Rocha, Distributed multi-robot patrol: A scalable and fault-tolerant framework, in Robotics and Autonomous Systems 61, 1572-1587, 2013.

[8] S. Gspandl, S. Podesser, M. Reip, G. Steinbauer and M. Wolfram, A dependable Perception-Decision-Execution cycle for autonomous robots, in Proc. of International Conference on Robotics and Automation, pp. 2292-2298, 2012.

[9] L. Lapierre and R. Zapata, A guaranteed obstacle avoidance guidance system: The safe maneuvering zone, in Autonomous Robots, Springer Verlag (Germany), pp.177-187, 2012.

[10] L. Jaiem, S. Druon, L. Lapierre and D. Crestani, A Step Toward Mobile Robots Autonomy: Energy Estimation Models, in Proc. of the 17th Towards Autonomous Robotic Systems, LNAI 9716, pp. 177-188, Sheffield, U.K., June 2016.

[11] M. Bennour, D. Crestani, O. Crespo and F. Prunet, Computer Aided Decision for Human Task Allocation with Mono and Multi Performance Evaluation, International Journal of Production Research, Vol. 43, No. 21, pp. 4559-4588, 2005.