AN EMPIRICAL EXPERIENCE WITH 3DROV SIMULATOR: TESTING AN ADVANCED AUTONOMOUS CONTROLLER FOR ROVER OPERATIONS

Daniel Díaz¹, Maria D. R-Moreno¹, Amedeo Cesta², Angelo Oddi², and Riccardo Rasconi²

¹Universidad de Alcalá, Alcalá de Henares, Madrid (Spain)
²CNR – Consiglio Nazionale delle Ricerche, ISTC, Rome (Italy)

ABSTRACT

The aim of this paper is to convey our experience using the ESA’s 3DROV planetary rover simulator as a visualization and validation tool through a dynamic analysis on the performance of an advanced autonomous control architecture: a power-aware, model-based control system with applications in rover-based mission operations which integrates advanced constraint-based reasoning within a flexible control execution management process. The analysis focuses on the validation of the main capabilities of the control system through three representative simulation examples executed upon an integrated testbed platform with 3DROV. More concretely, the target of the simulation tests is twofold: (i) to demonstrate the main capabilities of the controller; and (ii) to evaluate its performance on overcoming specific anomalous situations which might threaten the whole mission execution success.

Key words: Integrated testbed execution platform; mission plan execution management; dynamic validation analysis.

1. INTRODUCTION

Testing a planetary rover actually represents an important barrier on its design and development because of the lack of a complete knowledge of the target environment on which the rover is expected to operate, and the limitations on reliably reproducing the underlying conditions of the mission scenario itself. For this purpose, the “Planetary Robot Design, Generic Visualization and Validation Tool” [10] was conceived, simply known as 3DROV: an advanced software simulation platform mainly aimed at providing ESA’s Automation and Robotics (A&R) section¹ with a supporting tool for the complete development process of planetary robot systems.

¹The ESA’s Automation and Robotics (A&R) group is the responsible for carrying out with the creation and maintenance of such industrial technology base for the automation and remote control of space based operations.

One of the many interesting features of the 3DROV simulation platform is that it provides the capability to accurately model both the robotic system and the most important aspects of the environmental surroundings on which the mission is placed, as well as their physical interactions in virtually realistic scenarios.

In this paper we exploited such simulator capabilities through the development and deployment of an integrated testing workbench platform, with the aim of validating the basic target capabilities of the advanced model-based autonomous controller here referred to as CoRe³ which stands for (Co)ntrol (Re)sources & (p)ower (the main foundation concepts of the controller were presented in [4]) with application in planetary rover-based mission operations. The control architecture is targeted in generating and safely executing mission plans in a wholesome manner through a seamless plan execution management process implemented according to a single Sense-Plan-Act (SPA) closed-loop schema. More concretely, the execution management process is in charge of timely dispatching the rover operation commands that achieve the mission goals, while guaranteeing a flawless behavior by monitoring the execution progress so that the effects of possible disturbances can be easily detected and addressed. Reactive mechanisms close the execution loop by updating the internal model and triggering re-scheduling strategies if any misalignment is detected, so the whole mission integrity is always guaranteed. Additionally, high-level decision-making capabilities are provided at the core of the control management process through the use of an advanced reasoner [5] which is able of handling a wide range of complex temporal and resource constraints including power requirements [6].

The organization of the paper is as follows: in section 2, we give a general description of the problem scenario which we have considered as the rover-based mission of reference, as well as its formal representation according to a constraint-based schema. Section 3 describes the integrated testbed platform built on top of the ESA’s 3DROV planetary rover system simulator. In section 4 we conduct a formal analysis on the performance of the basic target capabilities of the controller through the dynamic simulation of some representative cases of study. In Section 5 we describe our particular experience on the
use of the 3DROV simulator as testing tool. Finally, a conclusion and lessons learned section closes the paper.

2. MISSION SCENARIO OF REFERENCE

We envisioned a mission scenario inspired on the Mars Sample Return (MSR) concept [1]: a lightweight rover-based mission aimed at acquiring Martian rock and soil samples from a set of scientifically interesting places to be later delivered back to Earth for a further analysis. This futuristic context justifies a smart rover equipped with advanced autonomous control capabilities. Future planetary exploration campaigns like the MSR will pursue increasingly challenging goals whose efficient attainment will require enhanced rover autonomy capabilities\(^2\) [9].

More in detail, in a conventional day-to-day mission unfolding, the rover is involved in deciding and executing a set of scientific experiments consisting of (see figure 1): (a) travelling to remote locations of scientific interest previously identified on Earth; (b) acquiring Martian soil samples by using a drilling subsystem (s/s) and a sample container to transport them; and (c) delivering them to a final location where an Ascent Vehicle (AV) will retrieve the collected samples from the rover by using a robotic arm and initiate the return trip.

Figure 1: Mission scenario representation with the main mission activities: navigation, sample acquisition (drill and sample storage) and sample retrieval

Next, we give a detailed description of the model which encapsulates the most relevant features of the problem of reference, as well as its formalization according to a constraint-based representation paradigm.

2.1. Problem model formulation

The problem of “delivering advanced autonomous capabilities on a MSR-like mission context” involves combining robust decision-making capabilities and flexible execution mechanisms within a seamless execution process:

1. Baseline schedule synthesis. The mission execution process starts with the computation of a baseline schedule which synthesizes the whole rover activity within a specific makespan (e.g., one Martian day, or Sol). More concretely, the rover is in charge of executing a set of experiments \(E = \langle Exp_1, ..., Exp_n \rangle\), where each experiment \(Exp_i\) consists of the sequence of activities \((Nav_{S,i}, Drill_i, Nav_{i,F}, Rel_i)\). \(Nav_{i,j}\) is the navigation activity, describing the long-range traversals between two different waypoints \(i, j\), generally representing two different scientific target locations, the initial location \(S\) of the rover (i.e., the position from which the rover starts the mission), or the final sample retrieval location \(F\) (i.e., the final position where the samples are retrieved by the ascent vehicle); \(Drill_i\) is the science acquisition activity, and consists of deploying a drilling s/s at a specific location \(j\) to collect and store a soil sample; finally, \(Rel_i\) is the sample retrieval activity, through which the collected sample is acquired by the AV at the final location.

2. Flexible schedule execution and contingency solving. Once an initial schedule has been synthesized, the rover proceeds to its execution by timely dispatching the set of (low-level) commands related to each rover activity\(^3\). The whole schedule execution relies on a SPA closed-loop control model (see figure 2), i.e., it is continuously supervised/monitored in such a way that as soon as a misalignment between the expected results and the real outcome is detected, a contingency solving strategy is deployed with the aim of recovering the execution to a consistent state.

The attainment of the mission goals requires the synthesis and the execution of a feasible schedule of all the rover activities, while synchronizing the use of a set of different mission assets (or resources) \(R = \{r_1, ..., r_n\}\), such as: a (1) drilling s/s \(D_i\) used for soil extraction operations; a (2) sample cache \(S_i\) (or sample container) capable of storing the collected science for transportation (up to a maximum number of \(C\) standard sized samples); a (3) locomotion s/s, which guarantees stable ambulation functionalities to reach a desired target; a (4) power s/s consisting of a battery \(B\), and solar array panels used to collect and store the solar flux energy; and a (5) robotic arm \(A_r\) mounted on the AV, necessary to recover the science collected by the rover.

\(^2\)Providing more autonomy to rovers also significantly improves the mission efficiency in terms of science return and operational cost savings, by cutting the time-consuming human communication/intervention cycles from/to the rover.

\(^3\)The execution of each rover activity implies its translation into a predefined sequence of low level commands which directly operate the rover actuators.
The rover activities are defined by means of their minimum/maximum durations, start/due dates, and the demands of one or more of the previous resources. In general, the successful execution of each activity is subject to the accomplishment of the following temporal, resource and energy constraints:

- **Activity execution times.** All experimentation activities have flexible durations in order to take into account possible non-nominal behaviours during execution. More concretely, sample collection (Drill) and sample retrieval (Rel) activities have a minimum duration equal to the time required to be executed under normal circumstances; while the duration values of the navigation activities (Nav) are lower bounded by the traversal times, i.e., time required to cross the path joining two different locations \(i, j\). A detailed definition of the traversal time is given next.

- **Traversal times** (\(t_{ij}\)) represent the minimum time span required by the rover to move between two different locations \(i, j\) according to a nominal behaviour (i.e., nominal rover’s linear speed, normal terrain conditions, etc.). Each path joining pairs of locations is considered as a sequence of straight segments, which were previously discovered as a result of a *traversability map* computation process. In general, we do not assume that traversal times satisfy the triangle inequality property, since the traversed paths might be computed according to multi-objective optimization methods [7] over a three-dimensional Euclidean space (e.g., according to the distance, geographical features, rover’s locomotion system configuration, etc.).

- **Resource demands.** The sample collection (Drill) activities demand the drilling \(s_h D_s\) and a specific amount of energy \(e_{exp}\), the navigation (Nav) activities demand a variable amount of energy \(e_{ij}\) as well as one storage unit of the sample cache \(S_r\) resource (if the traversal starts after performing an experiment), while the Rel activities require the robotic arm \(A_r\) of the ascent vehicle. The execution of the power consuming activities (i.e., soil extraction and navigation activities) require a certain amount of energy that has to be entirely available at the start execution time of each activity.

- **Energy constraints.** The battery \(B_p\) is assumed to be continuously charged at a specific constant rate \(\sigma_{charge}\) (unit energy/unit time) through the solar arrays until the saturation level \(B_{max}\) (Wh) is reached (once saturation is attained, all incoming energy is discarded). Additionally, the battery is not allowed to be used below a minimum usage threshold \(B_{min}\) (% of the maximum capacity) for safety purposes. All the previous domain problem specifications have been modeled following a Constraint Satisfaction Problem [8] (CSP) description paradigm (see [4] for modelling details). A produced plan corresponds to a feasible solution (and can therefore be executed) when it represents a schedule of rover activities where all temporal and resource constraints are satisfied, i.e., the execution of each activity does not entail any resource overconsumption, and all the domain time/distance relationships are consistent.

3. THE INTEGRATED TESTBED PLATFORM WITH 3DROV SIMULATOR

In this section we present the integrated testbed platform we create to validate the whole CoReP autonomous control architecture, with special attention to the following baseline requirements:

- **Domain model is correct and complete.** The constraint-based representation model must actually encapsulate all the significant aspects of the MSR mission scenario of reference.

- **Reasoning capabilities.** The controller returns feasible solution schedules which represent realistic estimations in time, resource usage and power demand when applied to the practical cases of study.

- **Uncertainty management.** The obtained solutions must be characterized by a certain degree of robustness, i.e., the capability to absorb temporal and resource variations at execution time, and/or to allow efficient reactive mechanisms against the occurrence of environmental disturbances.

Figure 3 illustrates the integrated workbench platform with 3DROV simulator. Next, we explain in detail each compound module which compose it:

---

\(^4\)Representation of the terrain which contains information about the ease of traversal of different regions according to diverse terrain features such as hills or obstacles.
Our autonomous control system CoRep was developed by using the APSI Timeline Representation Framework (APSI-TRF) [3, 2]: an advanced constraint-based software development environment which allowed us to represent our problem domain, as well as to implement and deploy the whole execution control system, specifically the decision-making capabilities implemented within the resource-driven reasoner ESTAp [6]).

Between the controller and the 3DROV simulation system we created a middle module as a functional layer with the following purposes: (i) interfacing both systems (i.e., the controller and 3DROV) through a TCP/IP-based communication relay, (ii) translating the dispatched rover activities contained within the solution schedule to the specific command sequences in the low-level format defined by the 3DROV system, and (iii) requesting and interpreting the telemetry data provided by the simulator.

The 3DROV simulation system has a modular and distributed architecture which we organized in Figure 3 around three integrated modules for simplicity: (i) the generic controller encapsulates all the required functionalities 5 to operate with all the rover subsystems (i.e., locomotion, drill, sample cache, etc.); (ii) the rover and environment models is a compound of accurate models representing information about a variety of rover and environmental features such as rover dynamics, kinematics, power, thermal, or terrain and atmospheric characteristics; (iii) the 3D visualization component is the front-end of the 3DROV simulator and allows us to track in real-time the evolution of the simulation execution through a 3D representation of the complete mission scenario.

Finally, we created an exogenous event generator (eGen) which allows us to “stress the simulation execution” in a controlled fashion by defining and injecting a set of exogenous events to model execution contingencies. Such contingencies are defined before the simulation starts, and are dynamically injected during the execution at specific instants in order to provoke disturbances on the mission execution course, therefore forcing the controller to react by suitably triggering the reactive mechanism policies 6. The eGen module interfaces with the 3DROV simulator so that the effects on the simulation are seen as changes on the rover behaviour or status.

More concretely, a contingency is characterized through the following parameters: (a) the type of exogenous event (i.e., temporal or resource); (b) the moment at which the contingency starts affecting the course of the nominal schedule execution $t_e$; (c) the specific time $t_{aware}$ at which the controller becomes aware of it (this instant is usually placed in time so as to occur after $t_e$); and (3) the kind of effects on the rover behaviour (i.e., abrupt or progressive). For instance, we might schedule a temporal event which causes a delay during the navigation between two different waypoints, so that the effects are observed as a reduction of the rover’s linear speed.

Next, we conduct a formal analysis on the performance of the basic target capabilities of the controller through the execution of three different simulation examples.

4. AN ANALYSIS ON THE PERFORMANCE OF THE EXECUTION CONTROL PROCESS

In the current section we will provide an evaluation of the performance of the execution control process through three different simulations executed upon the integrated testbed platform with 3DROV. The target of the analysis is twofold: (i) to demonstrate the main capabilities of the autonomous controller CoRep and (ii) to evaluate its performance against the possibly occurring anomalous situations that threaten the plan’s nominal execution.

Table 1 shows in detail the main setting parameters of the problem scenarios here considered, consisting of

---

3DROV implements a generic A&R control system based on the ESA’s A&R standardised development concepts and guidelines captured within the Control Development Methodology (CDM) [11] framework.

6It should be underscored that no knowledge whatsoever about the exogenous events is used during the production of the baseline scheduling solutions.
three different instances (denoted as $MSR_1$, $MSR_2$ and $MSR_3$) of our problem of reference, each entailing the synthesis and the execution of a feasible schedule composed of five scientific experiments. Their execution course will be disturbed by a set of unexpected exogenous events affecting some of the rover activities (i.e., the drilling activity and navigation activity).

It is worth mentioning that the steepness of the time scale and power consumption/production rates during the simulations were intentionally augmented to (a) collapse a full working time-frame into a few minutes of simulation, and (b) demonstrate the main controller capabilities with more clarity.

The set of contingencies injected during each simulation were implemented in terms of five different exogenous events $e_i$, where $i \in \{1, \ldots, 5\}$. All of them were randomly generated in a controlled manner in order to make them happen around specific instants. Three different types of exogenous events were considered:

- The rover gets stuck for time periods characterized by a duration of $n$ seconds, where $n \in [2, \ldots, 6]$, because of a wheel slippage during the execution of a navigation activity. Exogenous events $e_i$, where $i \in \{1, 3, 5\}$ belong to this category, and are scheduled to occur during the first, third and forth navigation activities respectively.

- The rover is deprived of incoming power for time periods with a duration of $n$ seconds, where $n \in [2, \ldots, 6]$, during the execution of a navigation activity. The power collection efficiency of the solar panels is appreciably reduced during this period of time (e.g., we model the shadowing on the solar arrays due to cloudy weather or the rover being shaded by a mountain). Exogenous event $e_2$ falls in this category and is scheduled to occur during the execution of the second navigation activity.

- The rover is deprived of incoming power for time periods with a duration of $n$ seconds, where $n \in [2, \ldots, 6]$, during the execution of a drilling activity because of the same reason than in the previous case. Exogenous event $e_2$ recreates this contingency and affects to the execution of the first drilling activity.

The minimum and maximum bounds of the exogenous event durations were carefully selected in order to show that the controller might respond or not to a contingency situation. Due to the solution’s flexibility, the occurrence of an exogenous event does not necessarily entails a reaction of the scheduler since this depends on the extent to which the new conditions affect the execution status, as the disturbance effects have to overpass a specific threshold (both temporal and in battery usage) in order to be considered by the controller as a dangerous situation. As seen in table 1, the controller checks the real execution results every second so that an anomalous situation is considered to be harmful (i.e., it causes a significant disturbance on the execution course) when the following conditions are satisfied:

- If the rover is delayed 3 meters (at least) while it is navigating between two different locations. For instance, if the linear speed of the rover is 1 m/sec., the controller will trigger a reactive strategy when the rover suffers a delay of at least 3 seconds.

- Similarly, an anomalous battery consumption will cause a reactive response if the amount of energy consumed triples (at least) the nominal values. For instance, if the controller estimated that the battery State of Charge (SoC) will evolve according to a specific energy consumption/production rate $\sigma_{\text{prod/cons}} = -0.5\%$ (i.e., the SoC level diminishes in half a point from one instant to the next) while the rover performs a drilling activity, and the real SoC curve evolves at 1.0% during 3 seconds by producing a cumulative loss of 1.5% (see figure 4).

![Battery usage profile according to a nominal (left) and an anomalous (right) rover behaviour](Image)

We chose the previous alarm-triggering threshold values on the basis of specific error margins considered during the definition of the durations and energy consumptions of the activities in the problem model.

Table 2 summarizes the details of the occurrence of the previous exogenous events as well as their effects on the course of each simulation execution. Information and metrics provided are the following: the specific instants at which the exogenous events occur ($t_{v}$); the time instants at which the controller detects the disturbances (if any) caused by the exogenous event ($t_{\text{aware}}$); the kind of reactive response by the controller on the acknowledgement of the disturbances (i.e., temporal propagation and rescheduling steps) as well as the number of responses (between brackets if more than one occurs); the makespan of the contingent solutions according to the comparative performance value $\Delta LWU\%$, which relates the quality of the solution with respect to the previous one (in the

$$\Delta LWU\% = \frac{mk_i - mk_0}{mk_0} \times 100,$$

where $mk_i$ is the current schedule’s makespan and $mk_0$ is the baseline schedule’s makespan.
Test problem description features

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
<th>Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num. Experiments</td>
<td>5 experiments</td>
<td>Third experiment has a completion deadline (time-window communication requirement)</td>
</tr>
<tr>
<td>Sample cache cap.</td>
<td>3 samples</td>
<td>Max. num. of soil samples transported concurrently</td>
</tr>
<tr>
<td>Linear &amp; rot. speed</td>
<td>1 m/sec. (linear) 0.4 rad/sec (rot)</td>
<td>Rover moves by performing straight, soft moves and standing rotations</td>
</tr>
<tr>
<td>Max. battery cap.</td>
<td>1400 Watts</td>
<td>Battery starts 90% charged with a min. usage threshold of the 20%</td>
</tr>
<tr>
<td>Power consumption</td>
<td>200 Watts (T) &amp; 150 Watts (E&amp;S)</td>
<td>T: travelling, E&amp;S: soil extraction and storage</td>
</tr>
<tr>
<td>Power production</td>
<td>15.02 Watts</td>
<td>Linear, constant rate (in average)</td>
</tr>
<tr>
<td>Execution feedback</td>
<td>Every second</td>
<td>The frequency at which the controller monitors the real execution outcome</td>
</tr>
</tbody>
</table>

Table 1: Configuration parameters for the test problems $MSR_1$, $MSR_2$ and $MSR_3$.

From the previous results it is worth to note that, in general, the response times by the controller on providing a contingent situation are quite reasonable if compared with the execution times of the rover activities and the monitor frequency, i.e., the global execution consistency is maintained in the face of the contingent situations.

For a further explanation of the control execution management process, we provide a step-by-step description of how the simulation execution of the second problem scenario $MSR_2$ evolves. Figure 5 shows the rover commands timeline extracted from the baseline solution schedule when the execution starts, as well as the battery usage profiles before and after the occurrence of the first contingency, respectively. It is worth to note also that the baseline solution maximizes the use of the sample cache (drilling activities located at WP$_4$, WP$_3$ and WP$_2$ waypoints are consecutively executed without releasing the cache in between). In this example, all the exogenous events trigger inconsistencies on the running schedule and the controller succeeds in providing a contingent solution which overcomes their effects.

- The occurrence of the first contingency causes both a temporal delay and a battery overconsumption significant enough to trigger the monitor alarms: the autonomous controller detects misalignments both in the temporal pace of the execution and in the battery usage (i.e., a power overconsumption) by comparing both the evolution of the expected position and the SoC values with the telemetry readings. Contingency effects are detected and reflected on the internal model of the controller by injecting and propagating two additional (temporal and resource usage) constraints. As a consequence of the previous propagation, the controller detects two inconsistencies in the battery usage profile (at the instants $t$ 590 and $t$ 1080 in Figure 5) respectively. As a result, a re-scheduling step is triggered by providing an alternative solution consisting of a partial reshuffling and delay of the pending activities to be executed.

Figure 6 illustrates the resulting timeline of the execution commands after the first re-scheduling step, as well as the battery usage profiles before and after the second contingency effects propagation.

- The second contingency causes a battery overconsumption during the execution of the first drilling activity. Similarly the controller detects misalignments between the expected and real evolution of the SoC values and updates its internal model by injecting and propagating an additional power usage constraint. As a consequence, a re-scheduling step is triggered as response of a violation of the maximum power limit (new overconsumption peaks are detected around the time instants $t$ 650 and $t$ 1420). An contingent solution consisting on a delay of the pending activities is successfully provided.

Figure 7 shows the evolution of both the command timeline and the battery usage profile resulting from the resolution of the second contingent situation.

- Similarly, the third, fourth and fifth contingencies causes battery overconsumptions during the execution of the following three navigation activities re-
Event specification | Effects on the simulation
--- | ---
**Event id** | **tء (sec)** | **tءaware (sec)** | Temp. propagation? | Re-scheduling? | ∆LWU% | Re-scheduling time (msec.)
--- | --- | --- | --- | --- | --- | ---
\(e_1\) | 2 | 6 | Yes | Yes | 22.49 | 639
\(e_2\) | 40 | 45 | Yes | Yes | 0.64 | 467
\(e_3\) | 105 | 108 | Yes | Yes | 3.11 | 614
\(e_4\) | 181 | 184, 187 | Yes (x2) | Yes (x2) | 0.62, 0.61 | 493, 505
\(e_5\) | 270 | - | × | × | - | -
--- | --- | --- | --- | --- | --- | ---
\(e_6\) | 3 | 6 | Yes | Yes | 22.57 | 770
\(e_7\) | 15 | 19 | Yes | Yes | 0.69 | 489
\(e_8\) | 82 | 85 | Yes | Yes | 3.32 | 650
\(e_9\) | 180 | 183, 186 | Yes (x2) | Yes (x2) | 0.66, 0.65 | 721, 723
\(e_{10}\) | 238 | 241 | Yes | Yes | 3.18 | 567
--- | --- | --- | --- | --- | --- | ---
\(e_{11}\) | 2 | 5 | Yes | Yes | 6.81 | 1111
\(e_{12}\) | 15 | 19 | Yes | Yes | 0.84 | 673
\(e_{13}\) | 88 | 91 | Yes | Yes | 0.84 | 812
\(e_{14}\) | 283 | 286, 289 | Yes (x2) | Yes (x2) | 4.02, 0.79 | 656, 704
\(e_{15}\) | 318 | - | No | No | - | -

Table 2: Exogenous events specification and simulation execution effects.

5. LESSONS LEARNED

3DROV simulator fulfilled all our expectancies as it certainly supported the validation of the main CoRe® controller’s target capabilities, by providing all the necessary functionality required. It actually posed a really good opportunity for us to test our control architecture with an external tool of such an stunning capabilities: 3DROV offered us a dynamic, complex environment (both from the visual point of view and the highly accurate physics models), as well as realistic telemetry data which allowed us to compare/contrast our estimates and assessing the correctness of our internal model and generated mission plans. Additionally, it helped us on understanding better many aspects of the planetary surface exploration domain in general, and in special those related to the current boundaries or the existing limitations on the rover operations in realistic scenarios.

However, main drawbacks we encountered on its utilization were related to the difficulties on establishing a communication relay for command dispatching and telemetry collection between the generic controller (embedded within the simulator) and the CoRe® system. This problem would be enormously simplified by providing a functional layer which wraps-up all the primitives for low-level rover operation through an standardized and usable interface of services.

6. CONCLUSIONS

In this paper we have presented an empirical experience on the use of the ESA’s 3DROV planetary rover simulation by performing a validation analysis of the
**ACKNOWLEDGMENTS**

Daniel Diaz is supported by the European Space Agency (ESA) under the Networking and Partnering Initiative (NPI) *Autonomy for Interplanetary Missions* (ESTEC-No. 2169/08/NI/PA). We want to thank all the support obtained through ESA-ESTEC, in special to its ESA's technical officer Mr. Michel Van Winnendael and Mr. Konstantinos Kapellos. This work was partially supported by the Spanish CDTI project COLSUVH, leaded by the Ixion Industry and Aerospace company.

**REFERENCES**


