The aim of this paper is to present the results obtained by the group “Human Machine Interaction” of the STEPS project during the first year of research. It describes the design of a working prototype, applied to an EVA Rescue Use Case, that validates the Predictive and Command System of the overall HMI architecture, capable to guarantee a sufficient level of situational awareness to the human operator by providing him/her with feedbacks about the tasks currently carried on and with information about both its internal status and the surrounding environment. A sliding autonomy approach was followed. Human machine interaction is bidirectional and each of the two actors can take the initiative for an interaction; complex tasks have to be represented in a structured way by decomposing a high level task into subtasks until to more primitive commands. In space exploration, it is necessary to integrate steps of on-line planning with parts devoted to the actual execution of the task, by providing the system with the capability of reacting to some contingencies, re-planning an action when possible or asking help to human in case of impasse. This approach can augment the operator control, and, by sharing Human and Robotic capabilities, it is possible to predict results and to achieve or negotiate a predefined goal reducing the Ground Support’s needs.

Nomenclature

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS</td>
<td>Artificial Supervisor</td>
</tr>
<tr>
<td>AS-PR</td>
<td>Artificial Supervisor of the Pressurized Rover</td>
</tr>
<tr>
<td>DEM</td>
<td>Digital Elevation Map</td>
</tr>
<tr>
<td>DS</td>
<td>Deliberative Services</td>
</tr>
<tr>
<td>EL</td>
<td>Executive Layer</td>
</tr>
<tr>
<td>EVA</td>
<td>Extra Vehicular Activity</td>
</tr>
<tr>
<td>FL</td>
<td>Functional Layer</td>
</tr>
<tr>
<td>HRI</td>
<td>Human Robot Interaction</td>
</tr>
<tr>
<td>HS</td>
<td>Human Supervisor</td>
</tr>
<tr>
<td>HSL</td>
<td>Human Supervisor Layer</td>
</tr>
<tr>
<td>IS</td>
<td>Interpretative Services</td>
</tr>
<tr>
<td>IVA</td>
<td>Intra Vehicular Activity</td>
</tr>
<tr>
<td>PCS</td>
<td>Predictive and Command System</td>
</tr>
<tr>
<td>PLEXIL</td>
<td>Plan Execution Interchange Language</td>
</tr>
<tr>
<td>PT</td>
<td>Predictive Tools</td>
</tr>
</tbody>
</table>

1 Ph.D. Student, CSPP, Laboratorio di Meccatronica, luca.carlone@polito.it
2 Researcher, Università di Torino, micalizio@unito.it
3 Temporary Researcher, Università di Torino, nuzzolo@unito.it
4 Ph.D. Student, Università di Torino, scala@unito.it
5 Human Factor/Ergonomic Doctor, Infrastructure/Transportation Systems, domenico.tedone@thalesaleniaiaspace.com
**I. Introduction**

STEPS is a three years research project, started in January 2009, and focused on Human and Robotic space exploration. For TAS-I and the Piedmont Aerospace District, it represents the opportunity to bridge the technological gaps and to consolidate the reached objectives in a common platform.

The paper discusses a methodology for controlling a planetary rover, supporting humans during space exploration missions, and investigates the adoption of predictive tools for helping a human supervisor in assessing alternative solutions when the rover is in trouble. Although teleoperation is often undesirable due to high operator’s workload and communication delays, the current fully autonomous systems state of the art is still insufficient to cope with threats and anomalies in the very demanding spatial domain; as a consequence, it is desirable to consider forms of cooperation between humans and the autonomous systems. In such a scenario the role of sliding autonomy becomes critical for trading off between efficiency and robustness of operation: on one side the rover is capable of executing some activities autonomously, on the other side, it can take advantages by the human expertise when some unexpected events occur or request a confirmation on the plan it has to execute. The rover is equipped with a control strategy that accepts high-level commands from the human operator, and computes the sequence of actions for accomplishing them. Since the real status may differ from the expected one, such a control strategy must be able to refine at runtime the course of actions by exploiting on-line planners and advices provided by humans. It defines conditional behaviours of the rover according to its degree of knowledge about surroundings, and its autonomy level. The possibility of having different degrees of autonomy (that may change over the time), allows the human operator to intervene at different levels either by submitting high-level tasks or by teleoperating the rover or by suggesting the way of performing the next set of actions.

The main aim of this work is to describe how the concept of the sliding autonomy can be applied in the EVA Rescue scenario described in the companion paper. We will use the rover navigation task for the purpose of explaining and testing the architecture of the companion paper, and for discussing how consolidated methodologies can be combined and properly extended in order to support the human robot cooperation. The approach has been validated through a first working prototype implemented using an open source language of NASA, called PLEXIL.

In Section II we introduce the notion of the sliding autonomy and the relative problem of the situational awareness, after which we describe the architecture of our system. The section concludes with a motivating example that will be used throughout the paper as an application domain for describing our methodology. Section III shows the implementation of the control strategy of the system which integrates capabilities of monitoring of path execution, online path planning and human robot synchronization. In section IV, we will describe the details of a deliberative service that is the path planner the rover has at disposal. The section ends with an introduction on Predictive Tools by which the human operator may validate the path the rover has to execute. Finally section V analyzes a running example of our solution.

**II. Sliding Autonomy and its Architectural Implication**

While in some domains the adoption of full autonomy is often desirable (i.e., the robotic agent is completely autonomous in performing its actions), not all the activities in a space exploration scenario are adequate to be carried on in full autonomous manner by a robot (e.g. a planetary rover). In fact, safety conditions impose the (virtual) presence of a human operator controlling the behavior of the rover. However, the scenario where the robotic agent is completely teleoperated may result very expensive in terms of communication and coordination processes that must be considered; so the adoption of a new form of autonomy appears necessary, in order to smoothly pass from a fully autonomous agent to a completely teleoperated robot according to the current contingency.

Literature on the topic reports several approaches that deal with this problem. Often these works introduce new terms to define the methodologies they propose, namely: *adjustable autonomy*\(^9\,21^\), *sliding autonomy*\(^20^\), or *interactive autonomy*\(^8^\). However, despite the name definition, the fundamental and shared idea is the same, that is to combine the advantages of the two extreme forms of autonomy (i.e. the fully autonomy...
and the complete teleoperation) providing the robotic agent with capabilities of adapting its autonomy. For instance it can switch to a lower autonomy level if it is unable to fulfill mission requirements, or conversely to maintain the control whenever the robot recognizes to be able to accomplish such a task. The multiplicity of situations to consider in a complex domain implies that there may be cases in which the fully autonomy is desirable and others (that the robotic agent can assess only during execution) that may enforce the robotic agent to request the human intervention. By this way the human can intervene only if necessary; as a consequence, he/she is not overburden of workload (as happen in continuous teleoperation) and possibly may be able to supervise many robotic agents at the same time. Adopting such a strategy the autonomy can hence “slide” in time, ranging from a maximum (the robotic agent is full autonomous) to a minimum (the robotic agent is pure teleoperated).

Actually the problem is more complex since: (i) we have not established who is up to decide when to start a switch of control with the consequent changing of autonomy (with “switch” we refer to the passage of control of the process, i.e. from the robotic agent to the human and vice versa); (ii) what are the levels of autonomy the robotic agent can take into account. As refers the first point, the possibilities are summarized in: system initiative (only the system decides to request a support), human initiative (only the human may interrupt the execution taking the control of the operation) and mixed initiative (both the agent and the human may have the initiative of switching).

As concerns the levels of autonomy, they depend on the capabilities the robotic agent has, and there may exist not a unique definition. Although the autonomy levels could be just two: teleoperated (lowest level) and fully autonomous (highest level), it is possible to define a spectrum of intermediate levels. In fact, in many cases, the human intervention is not necessarily limited to the teleoperation, but could also include high-level forms of cooperation with the robot, for example, the human operator can approve or disapprove plans proposed by the robot. These high-level Human-Machine interactions improve the effectiveness of the whole system. Of course, a major problem consists in developing Human-Machine interactions in a way they appear as natural as possible to the human operator (as for example, natural language and facial expressions); however, this issue is out the scope of the paper.

A. Situational Awareness

Every time a cooperation between the human and the robot takes place, there is a flow of information that must be considered. An ideal situation is when the human is always aware about which is happening (or happened) in the system, and about the current knowledge of the robot. However, such an ideal condition is often unreachable or very demanding in real-world domains; for example, communications in the space domain are in general characterized by limited bandwidth and narrow time windows. Moreover, providing the human operator with a huge amount of information could have a negative effect as it would be not informative at all. Therefore, it is essential to select the correct amount of information to share/send and establish what are the necessary/relevant pieces of information which allow the human to understand the actual status of the system. This issue is generally referred to as the problem of giving to the human the right level of situational awareness. Sellner et al. use a buffer containing both video and raw data information: when it is necessary such information are showed to the human, who is up to take the control of the robot in object. However, it is not clear what is the amount of the minimum indispensable set of information to send and which is the frequency of such communications. As the same paper of Sellner claims, the problem is still open because “there is an obvious tradeoff between performance and reliability: having more information and a larger buffer typically leads to a more accurate assessment, by increases how long it takes to attain situational awareness”.

While the works by Sellner et al. have addressed the problem of sliding autonomy during the execution of a given plan of actions, Bresina et al. have considered the sliding autonomy paradigm at planning level. In their work a human operator is in charge of supervising the daily plan of a rover on Mars adding/deleting actions during the mission definition.

The methodology we propose in this work is closer to plan execution, however, as we will see for some aspects, also the planning and especially the re-planning (in case of recovering) is involved. Since the sliding autonomy paradigm has very significant impact on the control architecture of the robotic agent we briefly introduce the architecture of our system before discussing in details the control strategy.

B. Functional Architecture

Significant efforts have been devoted in building an architecture supporting autonomy which is as independent as possible from the specific application domain. A commonly used architecture is the one
defined and developed by Alami et al.\textsuperscript{2}, which is based on a classical three tiers approach and supports a fully autonomous robot. In this paper we adopt the same general schema but with a little modification for the purpose of clarifying the role of the human operator, as shown in Figure 1.

Figure 1. HMI Sliding Autonomy Architecture.

A detailed discussion is described in the companion paper\textsuperscript{1}. Here we report a description of the fundamental blocks:

- the Robotic Layer is the lowest level of the system and interfaces directly with the hardware of the robot (actuators and sensors). This layer receives from the higher layers elementary commands, which are interpreted and translated into commands for the hardware, and provides Executive Layer with sensors reading;

- the Executive Layer is the intermediate layer and it is up to interpret high-level commands from the user and translate into the elementary commands for the robotic layer. In particular, when the user submits high-level goals, the Executive Layer has to find a plan in order to reach such a goal. To do so the Executive Layer exploits a number of deliberative services as, for instance, path planning. Moreover, the Executive Layer is also responsible for supervising the execution of the plan: the EL must detect anomalies (i.e., failures or deviations from what was expected) recover (when possible) from them, possibly by reevaluating whether the goals are both achievable and still relevant. Also in this case the EL can exploit its deliberative services;

- the Human Supervision Layer represents the main difference from the Alami’s architecture\textsuperscript{2}. This layer has been added in order to interact with the human user not only by allowing the user to submit high-level goals to the robot, but also by allowing the user to intervene during the robot activities and by assuring him/her with some form of situation awareness;

- the Deliberative Services provide high level reasoning tools to the Executive Layer. Currently, the main service the EL exploits is the path planning. This service can be invoked using several “modes” where each “mode” produces a path that optimizes a specific criterion (e.g. time, safeness, etc.). For details see Section IV-A.

Observing the HMI architecture it is possible to acknowledge a difference in the granularities and

\textsuperscript{1} American Institute of Aeronautics and Astronautics
abstraction levels of data exchanged between the Functional Layer and Executive Layer. Data acquired from sensors (at Functional Layer) often contains low level information, and, in general, FL does not produce abstract representations of such information. On the other hand the Executive Layer needs high level information for carrying out its functions (see Section III), including action planning strategies and the monitoring of the plan execution. In general EL, HS and deliberative services may need as input some pieces of symbolic/abstract information that should be synthesized from the low level data originated by the sensor modules. Activities of decision-making that are distributed among Executive Layer, services and HSL should work on semantic structures (e.g. discrete map, Cartesian positions) that shall be built up from the raw data (e.g. camera, GPS data, etc.). Analogously it is desirable that high level commands, issued by the Executive Layer, should be suitably translated in lower level commands directly executable by actuators. As matter of facts, the HMI architecture has been defined to have more levels assigned to the resolution of different abstractions of the problem following suggestions of Alami et al.\textsuperscript{2}. From the previous consideration it stems the importance of the interpretation modules introduced in the companion paper\textsuperscript{1} where we further distinguish the interpretation modules into sensor interpretation modules and command interpretation modules.

C. Motivating Example

In this section we introduce a scenario in the space domain which will be used throughout the paper for exemplifying how the sliding autonomy can be usefully exploited.

The scenario is set on the Mars surface, and the actors are a mobile robot (the “rover”) and two astronauts. One astronaut is engaged in an extra vehicular activity (EVA) mission (collecting terrain samples), while the other is located within a control station and controls the mission progresses; this second astronaut is therefore involved in an intra vehicular activity (IVA). We assume that the IVA and the EVA astronauts are always in contact with each other, and that the communication is reliable. In particular, the EVA astronaut can contact the IVA for requesting the help of the rover (for example when the EVA astronaut has problems for going back to the base). The IVA astronaut, which plays the role of the Human Supervisor in the architecture of Figure 1, catches such a help request and activates the rover by starting a EVA-rescue mission. In order to accomplish the mission, the rover has to reach the EVA astronaut and bring him/her back to the base station.

In pursuing its goal, the rover relies on the deliberative services it is equipped with, and on its autonomy level. Actually the rover’s autonomy level varies within a spectrum of autonomy degrees, and for each of them a specific “negotiation” with the IVA astronaut starts. The negotiation is more or less sophisticated depending on the specific desired autonomy. Of course, the aim of such a negotiation concerns the strategy to follow in order to reach the EVA astronaut, the autonomy levels and the possible interactions with the human will be discussed later on in the following section.

As a matter of fact, moving throughout the territory can be very demanding in terms of energy consumption and very hazardous since the domain is an open and dynamic environment. Initially the rover is equipped with a map of the territory surrounding the base, which is used to infer a path-plan for reaching the EVA astronaut. Such an initial map, however, is just an approximation of the real world, so it is possible that obstacles (e.g. rocks and holes) are not represented faithfully within the map or even completely ignored. Therefore the rover must be able to detect on-line (i.e., during the execution of the path-plan) unexpected events, and to re-plan suitable strategy in order to take into account acquired information. In particular, the sliding autonomy approach is tailored on the problem of establishing the next trajectory to be followed by the rover.

III. Control Strategy and Sliding Autonomy Implementation

The Executive Layer (EL) realizes the rover control strategy; thus it accepts high-level commands from the human operator, and computes the sequence of actions for accomplishing them. For example, in the scenario introduced above, the high-level command is “start a rescue mission”, and the plan inferred by the EL (through the invocation of a path-planning service) consists in an ordered sequence of locomotion actions, which leads the rover to reach the EVA astronaut. The EL is also responsible for supervising the execution of such a plan of actions, taking into account possible unexpected deviations from the nominal behavior. For example, it is possible that the map of the environment used to infer the initial path is not updated, and consequently a portion of such a path may result to be unfeasible (e.g. the terrain is more slippery than previously estimated). The EL has to detect these deviations and possibly to update its internal
knowledge (e.g., the map) about the environment.

In addition to that, the EL control strategy also models all the interaction mechanisms for interfacing with the Human Operator in order to support the sliding autonomy and the relative problem of the situational awareness. In the following section we first introduce and discuss the scale of different autonomy levels we consider, and then we detail how the control strategy is implemented by exploiting the hierarchical formalism of the PLEXIL language.

A. Autonomy Levels

Since the aim of the sliding autonomy paradigm consists in supporting human-robot interaction at different levels of abstractions, from the pure teleoperation (the lowest level) to the full robot autonomy (the highest level), it is fundamental to define what the intermediate levels are, and how the human-robot interaction takes place at each of them. In this way, the Executive Layer can combine the facilities provided by the deliberative services (e.g., path planner), and the expertise of the Human Operator (i.e., the IVA astronaut). The following table summarizes the different autonomy levels we will consider; for each level the table briefly describes how the interactions between the rover and the human are established.

<table>
<thead>
<tr>
<th>Autonomy Levels</th>
<th>Rover Behaviors</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>The rover autonomously computes the path to follow and asks the IVA for a confirmation until a predefined time (management by exception). If the rover does not receive an abort within the expected time interval, the rover assumes that the proposed solution has been approved by the IVA and starts the execution of the plan.</td>
</tr>
<tr>
<td>High/Medium</td>
<td>The rover autonomously computes the trajectory to follow and asks the Human Operator for a confirmation (management by consent). In this case the rover waits indefinitely a message (confirmation/abort) from the human operator.</td>
</tr>
<tr>
<td>Medium</td>
<td>The rover provides the human with a set of alternative paths, and asks the human to select one path in the set. Each path optimizes a specific criterion (e.g., fastness, safeness, etc.), thus the choice of the human is driven by the particular criterion he/she wants to optimize.</td>
</tr>
<tr>
<td>Medium/Low</td>
<td>The rover asks the human the criterion to be optimized during the path planning process. In any case, the human has to confirm the path the rover infers.</td>
</tr>
<tr>
<td>Low</td>
<td>The rover asks the human for a complete set of waypoint to be reached.</td>
</tr>
</tbody>
</table>

Table 1. Autonomy Levels.

The difference between the first two autonomy levels is that while in the highest the EL waits until a predefined time out, i.e. if the human operator does not answer the Executive Layer assumes the plan is valid, in the second level of autonomy the Executive Layer has to wait indefinitely. Such an approach is based on the concept of the management by consent and by exception.

The Medium level of autonomy implies that the Executive Layer has to invoke the path planner with a list of criteria. For each of these criteria, the planner has to find a path which optimizes that specific criterion. Therefore, the result is not a single path but a set of alternative paths that will be proposed to the human operator who, in turns, will answer with his/her best one choice. The management is implicitly by consent.

The medium/low level imposes the Executive Layer not to autonomously select the criterion to optimize in planning the next trajectory to follow. This autonomy level is taken into account for the purpose of combining both the rover capability (in finding a path) and the point of view of the human operator. For instance, the human operator may want to guarantee the safeness of the rover rather than minimize the time.

The lowest level is the one that represents the case in which the EL is not allowed to autonomously infer a path; therefore the EL invokes the human for obtaining the whole list of way-points.

For each level of autonomy, we assume that the target (e.g. the coordinate of an EVA astronaut) to reach is given by the Human Supervisor (i.e. it is not autonomously inferred) and whether the autonomy is set to be high, the desirable path is chosen according to a tradeoff among the possible criteria (i.e. no criterion is privileged); the actual implementation of the planning algorithm, how it allows to optimize some specific
B. Control Strategy Implementation

The control strategy of the Executive Layer has been implemented by exploiting the Plexil language and the Universal Executive API, used as interpreter of Plexil programs. The Plexil language has been conceived to describe tasks a robot can be assigned with in a hierarchical way (i.e., by means of a decomposition of tasks into sub-tasks).

The fundamental block of the language is the “node”. Each node represents a task and can be decomposed into a number of nodes (i.e., sub-tasks). The decomposition terminates when the leaves of the resulting tree are atomic nodes, which are either commands or function calls. Commands are directives for the robot actuators (i.e., commands are interpreted by the Robotic Layer which interfaces with the robot hardware); whereas, function call nodes are links between Plexil and external (software) modules (i.e., services in our control architecture). The adoption of command and function nodes mechanism allows the programmer to decouple the definition of the control strategy through the Plexil high-level terms from the actual implementation of the robot functionalities (either software or hardware). Note that, as we will discuss, we have used function nodes in order to model the communication between Human Supervisor and Executive Layer required to support the sliding autonomy paradigm.

The execution of a Plexil program starts from the root node which activates all its children nodes, which represent activities that are carried on concurrently. However, the programmer can specify gate conditions in order to rule the activation of a node both as a consequence of the conclusion of other nodes, and as a response to events captured by external modules (e.g., a node can be activated when the temperature reading captured by a sensor is greater than a predefined threshold).

![Figure 2. The High Level Control Strategy.](image)

The sliding autonomy control strategy implemented by the Executive Layer is shown in Figure 2. The “Control Strategy” node is decomposed into a number of nodes representing the high-level steps of the supervision task, namely:

1. “Flexible Path Computation” realizes the sliding autonomy behavior of the rover; in fact, this node is responsible for selecting the right strategy to infer a path relying upon the current autonomy level.

---

* Plexil is an open source language developed by NASA. The Universal Executive is a library for interpreting and managing the Plexil plan execution.

** Actually there are many other types of node we will not introduce for sake of simplicity.
2. “Execution” interfaces with the robotic layer by submitting the locomotion commands that must be executed.

3. “Monitoring” is the node in charge of supervising the execution of the locomotion, in particular this node has to detect the occurrence of anomalies in the execution and promptly react to them by activating a recovery strategy.

4. “Situation Awareness” is the node that manages the communication with the human user in order to provide her/him with pieces of information about what is happening.

As mentioned earlier, these nodes are concurrent processes; however their children nodes are actually activated according to the specified gate conditions. For example, the locomotion loop and the monitoring have to start once the new plan is inferred as well as the flexible path computation node has to start when a new plan is necessary. In the following, we will briefly consider each node and describe its relations with other nodes in the control strategy.

C. The Execution

This node manages the execution of a path, i.e., a sequence of locomotions. In particular, it has a single child-node, the “Locomotion Loop”, which is activated whenever a new path is available (see the starting conditions of the node). The “Locomotion Loop” node submits each locomotion command to the Robotic Layer (the Robotic Layer translates the locomotion into low-level commands directly executable by the hardware, see Section II-B). A locomotion, however, may fail; this happens when the Robotic Layer throws an exception (e.g. the left wheel is stuck), in this case the gate condition associated with “Locomotion Loop” node stops the execution of the plan, and throws the UnreachableWaypoint event which will be captured by the “Anomaly Detection” node, described later on.

The “Locomotion Loop” terminates with success when the end-condition statement is satisfied, in this case when the robot reaches the goal.

D. The Monitoring

The Monitoring node is subdivided into two subtasks: the former is the anomaly detection that constantly receives raw data coming from the environment, whereas the latter is the recovery task that is in charge for activating the flexible path computation node.

The anomaly detection node copes with the problem of catching failure or dangerous deviance from the original plan by evaluating the real effects on the world of each locomotion in respect to the prediction made. The monitoring is performed “before” and “during” the execution of each action:

- “during”, by evaluating the current real world data coming from the sensor. If such a new data represents a disagreement from the predicted evolution (made by the path planner) or a not safe situation, a not(GoodLocomotion) effect has thrown out with the consequent starting of a recovery and an abort to the current locomotion command (i.e. the invariant-condition of the current locomotion is no longer satisfied);

- “before”, by evaluating the action preconditions: when they are not satisfied (the real world may differ from the state expected at planning phase) the locomotion has to be prudently aborted: the monitoring throws a not(FeasibleLocomotion) event which is caught by the locomotion-loop node.

For both cases, whenever the “Anomaly Detection” node captures an anomaly, besides interrupting the execution of the locomotion, activates the “Recovery” node, which is responsible for correcting the current trajectory. To solve this task the “Recovery” node can exploit the deliberative services, as for example, the same path planner that has been used to infer the original path.

E. The Flexible Path-Computation

On the left of Figure 2 we have the activity that copes with the problem of finding the new set of locomotions for reaching the Eva Astronaut. For the sake of readability, Figure 2 hides a part of the decomposition which we show in Figure 3.

The “Flexible Path-Computation” node carries on the human-robot interaction according to the current level of autonomy: each subtask included in this node models a different way to obtain the set of way-points upon which the control strategy will build the sequence of locomotions. Note that in this case the decomposition does not consist in a set of concurrent processes where the activation depends on events generated by the other nodes, but the branching depends dynamically on the current autonomy degree.
Therefore, if the autonomy is “N” the strategy will follow the “N” branch and the other nodes are skipped.

As in the previous tasks of the strategy, the activation is ruled by specific gate conditions (i.e., start and skip condition). If a start condition holds the node is executed, conversely if the skip condition is satisfied the node is skipped. It is worth noticing that the start condition statement is the complement of the skip condition one (see gate conditions in figure 3).

Therefore, such conditions verify the autonomy causing an exclusive selection of the task that must be performed:
- when the autonomy is high the strategy selects the “query-for-confirmation” which will call the service of the path planning (the arrows are intended to make explicit the interaction between the strategy and the modules the architecture encompasses), and it will wait for the human acknowledgement since the human has the final decision. Indeed the confirmation node is split into two other nodes: one for the management by exception and the other for the management by consent (for sake of readability the figure hides such a further decomposition). The management by exception is implemented by means of a specific \textit{EndCondition} whose role is that of intercepting the timeout message. The management by consent is realized by not defining the EndCondition of the node, thus the leaf node which captures the acknowledgment waits indefinitely for the human’s response\(^6\).
- when the autonomy is medium the strategy opts for the query-for-selection node. The path planning is invoked by setting the “multiple paths” option (see details in Section IV-A), in this way the strategy has to take care of a multiple disjunctive set of possible alternatives (the fastest path, the safest...). The selection of the best one is left to the human.
- when the autonomy is medium/low the node to be activated is the query-for-criterion. Such a node invokes first the human supervisor layer in order to obtain the parameter the path planner has to optimize, and then activates the planner. After the planner provides the plan, the node re-invokes the human operator in order to obtain a confirmation (by consent) on the feasibility of the found path.

\(^6\) Indeed, the situation where a rover awaits indefinitely for a human response is quite undesirable. Consider for example a major fault in the communication system, which prevents the humans to contact the rover. If the rover waited indefinitely for a human’s response, it would waste time and resources. To avoid this situation we could set a timeout also in case of management by consent, however this timeout would be a long term deadline, after which the rover could rise its autonomy level and decide to start the rescue mission.
- when the autonomy is low the node to be started is the query-for-waypoints. So the path-planning procedure has to be skipped and the start condition of the query-for-waypoints is satisfied. The strategy will ask the human for the trajectory to follow.

In order to model and orchestrate via Plexil the invocation of the planner and the communications toward the human operator, the leaf nodes of the flexible path computation subtree are indeed function-calls that bind services and human supervisor layer together. Note that, once the path planner fails, or the human does not agree, the autonomy of the rover decreases until the lowest level of autonomy. On the other hand, the human operator can increment the autonomy level of the rover whenever the solutions proposed by the rover meet his/her requirements. For a clear example of this solution see Section V.

F. Situational Awareness

The situational awareness node (see Figure 2) copes with the problem of establishing what relevant pieces of information the human has to be provided with; in particular, the node is activated in one of the following cases:
- a locomotion has been successful executed
- a locomotion has failed for RL exception
- a locomotion has been aborted due at unexpected circumstances
- the target is reached

Note that these cases represent the set of start conditions, which are automatically checked by Plexil during the mission. The message that is sent to the human depends on the type of event that has been captured. For example, when a locomotion has been completed with success, the rover sends back to the human its current position. Whereas, in case an anomaly has been detected, the rover informs the human not only about its current position, but also about the specific anomaly that has occurred (e.g., an unexpected obstacle along the path).

In this way, since the human has received the plan of locomotions (in fact, despite the particular autonomy degree, the system always asks the human to validate the path), he/she can quickly understand where the rover is, and the path the rover has followed to get there.

IV. Deliberative Services and Predictive Tools for Rover Navigation

When dealing with model-based motion strategies we can identify at least three essential elements for autonomous navigation: (i) the knowledge of rover pose with respect to an external reference frame (localization); (ii) a representation of the surrounding environment (mapping); (iii) the capability of planning a strategy for reaching a predefined goal (path planning). Moreover, task execution can request further capabilities, ranging from manipulation, target identification and tracking, to mention just a few\(^{12}\).

Referring to the functional architecture introduced in Section II-B, we now analyze the services required by the Executive Layer in order to perform tasks, with application to the EVA RESCUE Use Case. In order to clarify the terminology we define a module as a computational unit able to process input data and provide desired information as output. The term is generic and hardware independent: processing can be performed by PC-based processors, FPGA, etc. They are able to provide services both on demand or with polling policies. We further distinguish two different types of modules:
- **Interpretation modules**: they can accept raw sensor acquisition as input data and provide integrated information as output, or, dually convert high level commands from the Executive Layer into low level commands to be applied to the actuators. In both cases the modules are characterized by a deep interaction with the Functional Layer. We define the services, provided by these modules, as **interpretation services**. These units are located in the Robotic Layer since they are interpreters of the data exchanged between the Functional Layer and the Executive Layer;
- **Deliberative modules**: they accept inputs from the Executive Layer providing **deliberative services** on demand. Their function is to provide the EL with decisional and planning capabilities (e.g. path planning, action planning, communication planning etc.).

It is worth noticing that there can be more than one module that provides the same service. For instance, it is possible to define two (or more) image recognition services using different algorithms: the first could reply faster than the second one, but with a higher level of uncertainty. The EL could choose the most suitable service according to how much time it can wait for the reply. In this way the system can trade-off among different factors, like the available resources or time at disposal.
A. Path Planning Implementation

The Executive Layer can request on demand deliberative services. The deliberative modules receive as input high level information (map representation, rover pose, etc.) and produce output data at the same level of abstraction. In this section we focus on the path planning service, showing how sliding autonomy concepts intervene in the planning process.

The Path Planning Module computes a path to be followed by the rover in order to reach a given target position from the actual rover pose. The path has to be optimal with respect to some cost function that, in our case, takes into consideration rover motion constraints, map geometry, terrain characteristics and energy consumption profiles. The map representation is assumed to be a digital elevation map in which the environment is discretized in a (regular) grid whose elements contain information about terrain elevation. Common planning algorithms model such grid representation with graph formalism in which nodes represent rover positions, whereas graph edges describes the cost of moving between contiguous locations. Hence graph-based search algorithms (Dijkstra, A*\(^\text{10,11}\)) can be naturally applied for planning purposes. Recent literature reports several algorithmic improvements which are mainly aimed at including non-holonomic constraints\(^\text{13}\) in the planning process or in order to produce smoother trajectories\(^\text{17}\). In this context we will not focus on the algorithmic aspects of planning, since we are interested in the interaction between the human and the robotic planner. In the rescue scenario it is assumed that a satellite map is known a-priori, hence we implemented the A\(^*\) algorithm which is suitable for planning on static maps; however we notice that the following considerations easily extend to other planning algorithms. A\(^*\) (A-star) algorithm is a typical example of best-first search algorithm; it adopts a heuristic for the estimate of the cost-to-go toward the goal node; this heuristic is always a lower bound of the true cost, so to guarantee the admissibility of the algorithm, i.e., to guarantee that the found path is optimal (lowest cost). The optimal path satisfies the following optimization problem:

$$p^* = \arg \min_j C(p_j)$$

(1)

where:

$$C(p_j) = \sum_{i=1}^{n} w_i f_i(p_j) = w_{xy} f_{xy}(p_j) + w_{Dy} f_{Dy}(p_j) + w_{Dj} f_{Dj}(p_j) + w_{Dq} f_{Dq}(p_j) + w_q f_q(p_j) + w_r f_r(p_j) + w_t f_t(p_j)$$

(2)

In (2) \(p_j\) is the \(j\)-th path (defined on the grid-based world model), \(w_i\) are suitable weights, and \(f_i(\mathcal{X})\) represent some functions of map characteristics and rover status information. In particular:

- \(f_{xy}(\mathcal{X})\) considers the length of the path to be traveled by the rover;
- \(f_{Dy}(\mathcal{X}), f_{Dj}(\mathcal{X}), f_{Dq}(\mathcal{X})\) are functions of the attitude changes (\(j\) is the roll angle, \(q\) is the pitch angle and \(y\) is the yaw angle) needed by the rover to reach the goal position;
- \(f_q(\mathcal{X}), f_r(\mathcal{X})\) are functions of the absolute roll and pitch angle and they are included to consider admissible attitudes, avoiding paths which violate rover motion constraints. Moreover the pitch angle is also used to distinguish areas in which the slope can be a risk for rover navigation, or simply to disambiguate upward slopes which are expected to increase energy consumption;
- \(f_e(\mathcal{X})\) is a function of the expected energy accumulated by the solar array during path execution. It includes a model of sun vector motion and considers the relative orientation of solar arrays with respect to sun position;
- \(f_t(\mathcal{X})\) is a function of terrain traversability: it is assumed that, in addition to the geographical DEM, a traversability map of the environment shall be available. It consists of a matrix having the same dimensions as the DEM, containing integer values \(t_{xy}\) which describe terrain characteristics. Some authors\(^\text{5}\) proposed the use of a traversability map, which is estimated from image processing from on-board cameras.

The model is augmented with information on energy consumption and recharge, terrain traversability,
attitude changes, allowing the planner to go beyond a naive distance-based criterion for optimization. In this way the robotic system may trade-off between different criteria, since the weights assigned to each summand in (2) can influence rover behaviour inducing it to prefer the minimum distance solution or different paths, which, for instance, maximize battery recharge or reduce undesired vibrations of the rover. In this phase of the project we consider the following planning modes, which can be chosen depending on the mission scenario, each one privileging a particular criterion in the optimization:

- **Minimum time mode**: cost of traveling long distances is stressed for the purpose of reducing the time required to reach the target. This can be the mode of choice in emergency situations, in which the rover has to reach the target quickly, eventually taking the risk of traversing uneven terrain (though respecting rover motion constraints);
- **Safe mode**: it allows the rover to reduce the vibrations in case an astronaut is carried on board the rover. Vibration are caused by fast changes in rover attitude or quick variations of rover elevation;
- **Energy saving**: the path minimizes energy consumption by taking into account upward slopes and traveled distance contribution to energy consumption. Moreover the estimated battery recharge is considered for the purpose of computing the optimal path;
- **Terrain adaptive mode**: planning takes into account the danger of traveling on terrain where slippage phenomena may occur;
- **Standard mode**: the weights are chosen so that a tradeoff among the criteria is accomplished (no criterion is privileged). This tradeoff is accomplished by a suitable setting of weight which is determined experimentally in practice and it is the mode of choice when mission scenario does not impose other criteria of optimality.

**Figure 4. Path planning example.** *Path alternatives produced in minimum time mode (red) and safe mode (blue), respectively. The two sub-figures on the right report the pitch angle expected during path execution.*

The previous modes give the rover the capability of producing different plans in function of mission or user requirements. The planner can also produce more alternative paths enabling the EL to make the human supervisor choose the preferred solution, thus enhancing human-robot cooperation in planning, as discussed in Section III.

In order to enhance situation awareness of human supervisor, the path planning service is expected to
provide, beside the waypoints the rover has to follow, some further information related to the optimal path, which can help the HS to decide the suitability of path execution. Such information can be provided at interface level in addition to the computed trajectory which is shown over the DEM of the environment, see Figure 4. Among the other relevant data, we considered the estimated time for reaching the goal, the maximum attitude angles, expected during path execution, the expected energy consumption and recharge. As described in the following section, it can be difficult for the user to take the decision on path execution, hence further tools for supporting decision making can be provided.

B. Predictive Tools for Supporting Decision Making

The framework proposed in the previous section is based on a flexible planning strategy which accommodates different autonomy levels. It can be easily understood that an improper choice of the level of autonomy can compromise mission execution: if full autonomy (autonomy level set to high) is assigned to the rover plans the path and executes it after a given interval of time. If the rover is not capable of correctly executing the task (improper autonomy level selection), the rover performs the task in a misguided optimism, increasing the risk of mission failure. In our current implementation this level is fixed by the human supervisor, hence it is fundamental that a proper setting of the autonomy level is assessed. The human supervisor may have low consciousness of rover capabilities and this can lead him to assign the rover a low level of autonomy, overestimating human capability in managing the task to be executed. According to such observations we are currently investigating the possibility of providing supporting tools for the HS, which can enhance his situation awareness and provide him with the objective information needed for decision making. The study is devoted to an application of such predictive tools to path planning procedures. In this case the tools provide data that can assess some confidence index on the path computed by the rover. In this sense they can be seen as a validation tool for the plan, whose results can help the HS in assessing the autonomy level of the rover. Apart from the aforementioned path information, reported in Table 2, we considered two main predictive tools: the static path validation and the dynamic path validation.

<table>
<thead>
<tr>
<th>Path information</th>
<th>Data description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimated time to goal</td>
<td>Time required for covering the computed trajectory under the hypothesis of rover constant speed</td>
</tr>
<tr>
<td>Estimated distance to goal</td>
<td>Sum of the Euclidean distances among the waypoints for the path</td>
</tr>
<tr>
<td>Pitch angle information</td>
<td>Maximum values and (when required) plot of estimated pitch trend. Values that exceeds given thresholds are suitably labeled and displayed on the interface with given color code</td>
</tr>
<tr>
<td>Roll angle information</td>
<td>Maximum values and (when required) plot of estimated pitch trend. Values that exceeds given thresholds are suitably labeled and displayed on the interface with given color code</td>
</tr>
<tr>
<td>Estimated energy consumption during path execution</td>
<td>Rough estimate of energy required for covering the path. The path consider slopes of the terrain to be traveled as well as motion commands</td>
</tr>
<tr>
<td>Estimated solar array recharge in function of selected path and sun position</td>
<td>Rough estimate of energy accumulated by solar arrays. The estimate is computed in function of the angle between the sun vector and the normal to the solar arrays</td>
</tr>
</tbody>
</table>

Table 2. Path statistics provided to the HS.

The first tool can be required by the human supervisor and accepts as input a potential trajectory for path execution. Such post processing on the candidate path is aimed at having a rough estimate of the reliability of the estimated path. Since the A* finds the optimal path for reaching a given goal position, such predictive tool is required to evaluate the sensitivity of the optimal path to perturbations of some parameters that characterize path execution. We considered two main parameter perturbation effects:
Path following errors with relative (odometric) localization: rover localization is based on (visual) odometry, hence localization error accumulates over time, increasing with the distance traveled. As a consequence the path following error (distance between the desired position along the planned trajectory and the actual position) is expected to increase over time. The accuracy of odometric position estimation can be modeled by means of covariance matrix referred to the rover pose. In several field robotic applications such covariance matrix is already available, or it can be computed using a suitable odometry motion model, since it is used for sensor fusion (for instance in the integration of IMU and visual odometry measurements) within the framework of Kalman Filtering. Therefore a covariance matrix can be associated to each waypoint of the desired path, with the meaning of uncertainty of the outcome of path execution. In such a case the static path validation tool is required to simulate the expected cost (as defined in A* algorithm) for paths which are perturbed (with respect to the desired trajectory) according to the uncertainty in the motion model, expressed by the corresponding covariance matrices. In practice the odometric estimation of rover motion is computed by adding random noise to the desired rover motion, where the noise distribution is compatible with odometry uncertainty. Then path sensitivity to perturbation can be evaluated by means of Monte Carlo runs over the uncertain motion, and for each experiment a cost value can be retrieved using equation (2). The distribution of such costs provides useful information of execution outcome. For instance, if the expectation of the costs obtained from the Monte Carlo runs remains close to the cost of the desired path, the planned solution can be safely executed. However, other statistics can be performed; for example, it is possible to analyze the number of experiments that produced unfeasible paths. In addition, more efficient approaches can be implemented, see Martinez-Cantín’s work and the references therein;

Path following errors with absolute localization: rover can obtain at each sampling period an accurate estimate of its absolute position (with respect to a given reference), which can be provided by satellite measurement or by beacon-based radio triangulation, for instance. Hence the path following error is expected to be bounded, with error bounds that are function of the localization accuracy and path following strategy. This case can be formulated as in the odometry-based localization, with a suitable choice of the covariance matrices.

For the first demonstration in the STEPS project only the second approach was taken into account. Moreover, since the state space (possible rover positions) is reduced in function of DEM resolution, the perturbed paths have to be discretized according to the grid representation.

The static path validation is able to catch kinematic aspects and errors that may occur during path execution. However dynamic aspects cannot be neglected when the rover travels at a relevant speed and on uneven terrain. For this purpose, if the HS needs a more reliable decision support, we further considered the dynamic path validation, which includes dynamic effects and other constraints, as wheel slippage, limited steering angle, non-holonomic wheel constraints, errors due to path following control algorithms, etc. When the Human Supervisor requires such a tool, a realistic simulation is performed, and statistics on execution results are displayed to the human supervisor. After a proof of concept in Blender for Robotics, the simulator of choice was the software tools adopted in STEPS project, namely TruckSim and RoSi. TruckSim is a commercial tool for simulating and analyzing the dynamic behaviour of medium or large size vehicles. Simulation can be also customized using external programs as Matlab and API interfaces. RoSi (Rover Simulator) is an application used to simulate and place the rover on the planetary surface. Both tools also provide a video streaming of the expected path execution, which can be displayed to the Human Supervisor via GUI, if requested.
V. An Example of Application

In this section we provide an example of application of the proposed framework to the mission scenario described in Section II-C. It is worth noticing that such example constitutes a simplification of the EVA Rescue Scenario proposed in the companion paper\(^1\), and it is used here for sake of simplicity. In our implementation the environment was simulated in Blender, and the DEM was modeled with a 3D modeling tool. The satellite map of the mission area is reported in Figure 5. In our simulated experiment the Executive Layer and the deliberative services were implemented on separated computational units, exchanging information via the Communication Layer. Path planner is able to send a response to a planning request in few seconds, whereas the communications delays were neglected, assuming that interaction takes place only between agents on the planet surface (IVAC in control station) and the small rover. The CL was responsible for information exchange and it was based on a client-server architecture. We remark that these preliminary results are reported here for reader convenience, in order to clarify the proposed framework with a simple example of application, and they are not intended to be an experimental validation in a realistic context.

The small rover (A) is performing an exploration task and its autonomy level is medium-high, see Table 1. In the meanwhile the EVA is in position (B), looking for scientific sites of interest. The control station, which in our case study is a pressurized rover\(^1\), is labeled with (D). For sake of simplicity the small rover will be referred to as rover, whereas the pressurized vehicle is considered as a fixed base station. During standard operation the astronaut health monitoring system reveals an anomaly in physiological parameters of the EVA astronaut and the IVA operator (later referred to as Human Supervisor) issues the rover to start a rescue mission. Rover invokes the deliberative service and, according to the autonomy level, the planner provides it with the optimal path; then the EL waits for a feedback from the human operator. The HS makes use of the static validation tool, whose response assures the reliability of the computed solution. Therefore the IVA astronaut accepts the proposed path, starting the execution. The planned path is shown in red in Figure 6. During path execution the rover moves from position A to position C, where it encounters a non-traversable zone. Rover sensors allow inferring the size of the non-traversable area, hence the rover can safely re-plan its trajectory, in order to reach the goal, avoiding the stay-out zone, see Figure 7. When the rover reaches the EVA astronaut, the latter is carried on-board and the rover is required by the human supervisor to come back to the Pressurized Rover site. The EL of the small rover recalls the path planning services, obtaining a new path solution, which is sent to the HS for taking the decision on execution. The Human Supervisor uses the dynamic validation tool, in order to have stronger guarantees on the outcome of the path. The predictable tool reveals that some motion constraints are violated during path simulation; moreover the HS requires the plot of the expected rover attitude, and it shows how the rover, thus the astronaut, will traverse uneven terrains, with risk for EVA crew safety. The HS decides to decrease rover autonomy level, since the planner is not able to correctly solve the problem: the autonomy level is fixed to medium, see Table I. The EL invokes again the path planner, which now provides different path alternatives to be displayed to the HS. The operator chooses the path corresponding to the safe mode, but prefers to use again the dynamic validation tool, in order to check path reliability. The tool confirms the safety of the computed

![Figure 6: EVA Rescue Example. Path computed for reaching the EVA astronaut (B), starting from the Small rover position (A). Position D corresponds to the location of the Pressurized Rover.](image)

![Figure 7: Re-planning. A non traversable zone is detected when the rover is in position C. The re-planned path is shown in blue.](image)
trajectory, and the HS can send a positive feedback to the Executive Layer for path execution. The rover follows the computed trajectory, finally reaching the Pressurized Rover, where the disabled EVA astronaut can be helped and the normal working condition reasserted.

**Conclusion**

The increasing request of autonomy for planetary rover stresses the importance of overcoming purely reactive behaviors in order to enhance the decision making capability of the robot exploiting the perception of the outer environment. The maturity of field robotics enables such autonomy although the interaction with human actors remains crucial in order to fulfill mission requirements. The possibility of adapting the level of autonomy of the robotic system (sliding autonomy) and the negotiation process between human and robotic control have attracted large attention from recent literature. In this work we report first accomplishments in the application of sliding autonomy concepts to the “Human Machine Interaction” group of the STEPS project. The paper briefly reviews the state of the art on sliding autonomy and proposes a functional architecture enabling human-machine interaction with different levels of autonomy. The scheme, which is further discussed in the companion paper\(^1\), allows a bidirectional flow of information among the robotic and the human agents and is able (i) to provide the human supervisor with mission status information, in order to enhance his situation awareness; (ii) to give monitoring and on-line planning capabilities to the Executive Layer of the rover. The implementation of the sliding autonomy is then presented and detailed, with application to the path planning problem, which arises when dealing with the EVA Rescue Scenario, chosen as a test bed for the first demonstration in STEPS. The framework defines different rover behaviors for each autonomy level and intrinsically encompasses rover capability of reacting to some contingencies or asking help to human in case of impasse. The path planning service is tailored on Executive Layer requirements and allows a flexible planning strategy that can improve human-robot cooperation during planning. The work also introduces some Predictive Tools which are intended to be a support in human decision making, with application to the definition of plan confidence indices. Such indices are particularly relevant for a proper setting of the autonomy level which turns out to be a main prerequisite for correct mission execution. The proposed framework is finally applied to a practical example, which shows the potentiality and the flexibility of the sliding autonomy scheme, and the application of the aforementioned Predictive Tools.

Future works include the extension of the sliding autonomy approach to other deliberative services in order to provide a more general scenario of human-robot cooperation. In particular the interaction between the Executive Layer and Services will be considered for the purpose of monitoring astronaut’s physiological parameters, and its implications on autonomy level definition will be evaluated. Research effort will be also devoted to the implementation of a target tracking system which allows a shared control strategy between the human and the robotic agent.

**Acknowledgments**

We would like to thank the STEPS Steering Committee, Thales Alenia Space Italy and Piedmont Region for co-founding the research project, the TAS-I team that is working in the project, the Universities (Università di Torino, Politecnico di Torino, Università del Piemonte Orientale), the Aerospace Department of the “Politecnico di Torino” and all the Small Enterprises that are working to the project (ALTEC, AMET, NEKHEM, SKYTECHNOLOGY, SEAC02, TEORESI, Sofiter System Engineering). We would also like to thank Eng. Enrico Gaia from TAS-I Torino.

**References**

*Periodicals*


**Books**


**Proceedings**


**Computer Software**


25TruckSim, Software Package, Ver 8.0, Mechanical Simulation Corporation

26PLEXIL and the Universal Executive Ver 1.0, http://sourceforge.net/projects/plexil/