ON-BOARD SIMULATOR FOR AUTONOMY ENHANCEMENT IN ROBOTIC SPACE MISSIONS

Domínguez, Raúl[1], **Schwendner, Jakob**[1], **and Kirchner, Frank**[2]

[1]*DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany*
[2]*DFKI GmbH, Robotics Innovation Center and University of Bremen, Robotics Research Group, Robert-Hooke-Straße 1, 28359 Bremen, Germany*

ABSTRACT

Space robotics missions are subjected to hard challenges on deployment time and the higher the level of autonomy the larger the magnitude of these difficulties. Its final behavior is always dependent of the current states of the environment and of the system itself. In order to provide the required level of reliability, accurate predictions of these behaviors is mandatory for planning. This document proposes an approach based on the use of complex simulations involving the models of the environment, of the robot and of the whole control software as a tool to provide this predictions and improve the available planners without modifying them internally. The approach pursues affecting the planner's behavior through different external means (e.g. modification of it inputs, parameter adaptation) so that it will produce more reliable solutions. An application for improving the efficiency of a naive planner aiming to solve a salesman travel mission scenario is presented. The connections between the different locations to visit are assumed by the planner to be connected but the final scenario and the navigation limitations of the robot do not allow the traverse of all the paths. Through a simulation based on an aerial image the valid paths are found and a plan is generated which is not leading to a failure state.

Key words: On-Board Simulation, Internal Simulator, Planning, Autonomy, Space Robotics, Validation, Forward Models.

1. INTRODUCTION

Space exploration demands robotic systems that perform reliably and autonomously complex missions in unstructured environments (e.g. exploration of caves in Mars). As the demands on autonomy increase, the complexity of the software that control the robot does so, as well as the physical complexity of the system. The combination of these three factors: Behavioral, dynamical and environmental complexity pose a hard challenge when aiming for a reliable and intelligent system. A key factor for achieving operational safety is the capacity of the system to predict with realism the outcomes of its actions. It is here proposed an approach in this direction based on highly complex physical simulations in which it is pursued to reproduce the whole system's behaviors mirroring as much as possible its real execution.

The concept of the *Internal Simulator* is present in cognitive science theories to explain high level cognitive activity (thinking)[Bar99, Hes12] . These theories have inspired robotics research where theories about imagination functions are proved functional[MH09] and useful. On the other hand, Artificial Intelligence literature exists in which learning from an internal model and not only from reality enhances efficiency [SB12] and applications of the same principle have been proposed in the manipulation field [Mel88]. Some examples already exist where on-board simulations are used in complex robotics simulations of space missions [RmGA$^+$14]. Though, in the literature real missions applications with complex robots are scarce. More applications are envisioned in our group, mainly related to in operational safety and autonomy. Two crucial aspects in space missions because communications might be impossible for long periods and error costs are very high.

An experiment aimed to point out is the capacity of the Internal Simulator to improve available modules without the need of human intervention is presented. In particular, the experiment shows two features that an Internal Simulator can be useful for: efficiency enhancement and design limitations overcoming.

In the experiment, a planner of a simulated robot will enhance its efficiency in solving a salesman problem with non traversable paths. Initially, these non traversable paths are assumed traversable by the planner and thus proposed as valid solutions that fail at execution. The Internal Simulator will use additional knowledge of the environment (a 2D overhead image of the area) to generate and execute simulations of parts of the mission. Detecting in advance the failures and avoiding them (see [VZ06] for a similar experiment). Furthermore, the knowledge obtained from the simulation will be integrated in a structured representation that will be integrated along the original planner. Eventually, it will not be necessary to simu-

late in order to determine whether a path is traversable or not.

Thanks to the advances in computer science and engineering it is now feasible to integrate complex simulators in the control software architectures of robotic systems. Meanwhile, the physics simulation engines and the simulation models are increasingly efficient and realistic. For instance, models for terramechanics simulation is an active research area in the field of space robotics. In this work, it is aimed to study the applications that an onboard simulator can have in space missions performed by robotic systems. Reliability, efficiency and autonomy enhancement are some of the envisioned uses.

In Section 2 the ideas on how to integrate the tool in the robotic controller are explained in detail. In Section 3 an illustrative experiment is presented in which a robotic system enhances its chances to success on a navigation mission thanks to the Internal Simulator. Finally Section 4 summarizes the most important ideas and explains the next research objectives.

2. PROPOSED APPROACH

For introducing the concepts a first brief look on the cognitive paradigm in which the Internal Simulator is introduced from a functional point of view is presented. Then in the Second part of this section a more close to robotics engineering vision of the approach is provided.

2.1. Cognition and the Internal Simulator

The Internal Simulator is a high level tool, which can perform different tasks. In general, it brings robustness and safeness to the system but it can also provide a tool for learning new behaviors. Following the Levels of Behavior model of cognition [KRS+12] (Figure 1), the Internal Simulator takes the highest position, here the challenge is to detect failures and limitations in the plans and the models, explain them and apply this knowledge to improve the system.

The cognitive hypothesis is that some intelligent behavior is performed *automatically*, when for instance the task is repetitive or known. But to achieve some tasks an expansion of this *automatic* thinking is often required and there is where the Internal Simulator as highest cognitive feature is used. Its task is to simulate acting, consequences and furthermore learn to predict and adapt with this knowledge the *automatic* thinking.

This idea is being brought to the field of space robotics. The planners and models that the system uses for performing a task must reduce the complexity of the space of search (i.e. the environment model) to a computably feasible one. Furthermore, they may not account with the whole interaction of other components (e.g. reactive

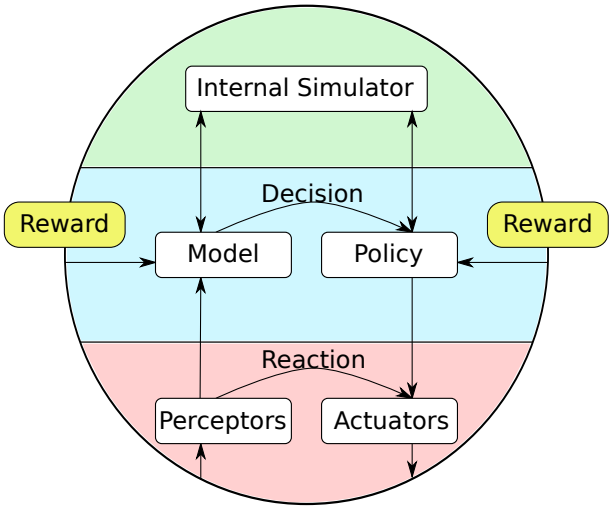


Figure 1: The Levels of Behavior Model is chosen as the architectural paradigm upon which the modules of the robotic system are presented. In this figure, the area outside the circle represents the environment.

behaviors). Thus, the space for possible failures in execution time becomes large. On the other hand, simulations can represent the reality and the robot itself with its all complete behavior accurately but its computational cost and the high dimensionality of the space of search makes its use inefficient for planning. The Internal Simulator that is envisioned pursues to get the best from both elements by validating the plans and performing only expensive simulations in the cases where the plan success is not guaranteed.

2.2. Modular Robotics and the Internal Simulator

The Internal Simulator is therefore closely related to the Planners, the Models and finally to the Environment representation. This last component includes more information than those present in the Models about the environment and the system itself but it would be impossible to account with it all for real time planning in general. On the other hand, when a limitation on the system is detected some solution might be possible if the information included in this environment were taken into account.

An Internal Simulator can affect the behavior of a planner in several ways. Here we present an example in which by changing the order of the goals, it is achieved the generation of valid solutions. Other manners in which an Internal Simulator could affect the planner is by modifying its policies or by introducing non existing elements in the environment representation so that the generated models for planning encode this artificial information (e.g. [CM09]).

For an Internal Simulator to provide useful results four main requirements are identified:

- Generative: Capability to generate, from the exist-

ing information about the environment a simulation environment.

- Executive: Capability to execute, based on the particular constraints of the mission to be validated, the necessary simulations.
- Adaptive: Capability to alter the existing software so that the information from the simulation can be integrated in the execution loop.
- Supervisory: Capability to supervise the execution of the mission either in real world and in simulation to enable the detection of failures and successes,

Given these requirements the Internal Simulator provides the tools to detect the regions in the space of possible solutions generated by the running components which either for limitations on the software side or hardware limitations will not produce the desired results.

- Software that attempts solutions which can not be physically executed.
- Software that attempts solutions which are not desired by the designer.
- Software and hardware that generate an undesired behavior.
- Hardware features that are not exploited by the software and that are not desired.
- Hardware features that are not exploited by the software and that are desirable.

3. EXPERIMENTS

The experimental results here presented were implemented using the *Rock, the Robot Construction Kit*[Roc] and the simulation environment *Mars* [Mar]. Rock is a framework for programming and controlling robotic systems. It is based on the principle of having modular components (i.e. tasks), each with an specific function that are connected through ports. Mars is an application for simulating robots and the environment where they are inmersed, it is integrated in the Rock framework but can also be used independently. Mars internally uses Open Dynamics Engine [ODE] for computing the physical interaction between the components of the simulation.

To exemplify the potential capabilities of the approach, a case of the classical salesman problem is presented. The robot has the mission to visit certain positions of an unexplored area, of which aerial imagery is available.

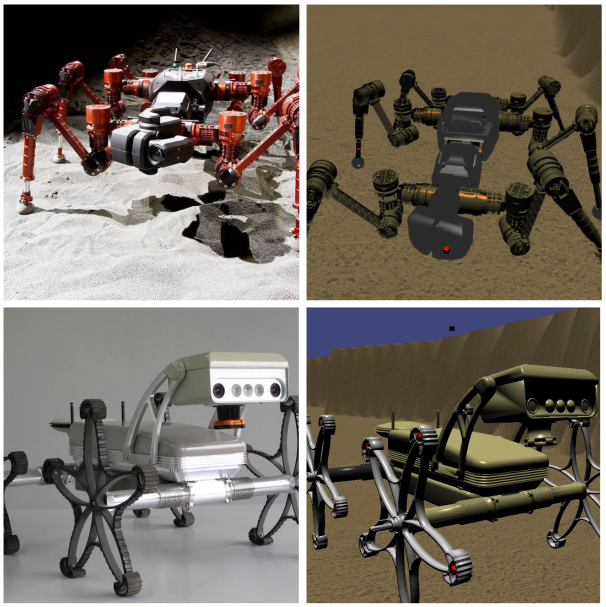


Figure 2: The experiment was performed with two robots of different morphologies. In the picture the real and its correspondent simulation are shown.

3.1. Robotic Systems

The experiments were performed with two simulated robots. Both robots are equipped with a laser sensor for environment perception. *Crex* robot is a 6 legs robots designed for negotiating complicated surfaces such as caves and crater surfaces. The *Asguard* robot has a combination of wheels and legs, that are more energy efficient than legs and allows the traverse on more unstructured surfaces than wheels (e.g. rocks surface).

The components that enable the robot to perform the navigation task are depicted in the diagram in Figure 3. The localization is performed in this experiment based on odometry. There is no slam algorithm or global model of the environment. The perception of the environments is based on the data provided by a tilting laser sensor located at the front part of each robots. The sensor produces a pointcloud from which a local traversability map is built. This map is then converted to a grid upon which the local planner plans a trajectory avoiding the obstacles. The local trajectory pursues to follow the global orientation which is a straight vector between the current position and the target position.

The environment representation module uses data from different sensors such as laser sensors and inertial measurement units to generate a model of the environment. This module also incorporates the geo-referenced aerial image. The environment representation is used by different tasks that generate models in which the planning algorithms search for action chains in order to arrive to the goal state. Furthermore, the Environment Representation module includes external information about the environment where the mission will be performed. For this case

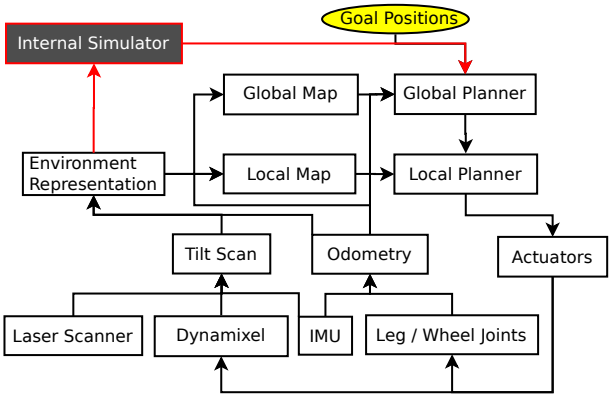
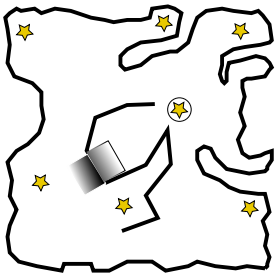


Figure 3: Modules that control the navigation of the robots. The Internal Simulator uses the Environment Representation to generate the simulation environment. The Internal Simulator overrides the goals input to the global planner. In this way, only validated outputs will be generated.

a post processed aerial image is available from which a simulation can be generated (see Figure 4).



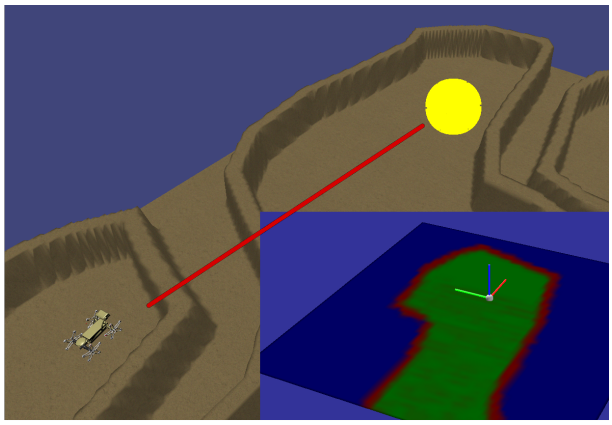
(a) Possible post processed aerial image of the deployment scenario. The stars represent the positions to visit. The circled one is the start position. The gray areas correspond to heightmaps.

(b) Automatically generated simulation scene. In Mars[**?**] this 3D object is used as terrain object.

Figure 4: From the available information about the environment, the Internal Simulator should generate a realistic as possible simulation.

3.2. Problem Description

The goals are provided to the global planner which determines using a global model the global path to follow. The limitation that is proposed here to overcome with the Internal Simulator is one of the global model. This task assumes that all paths are traversable but reality is different. The system can get stuck in certain regions due to characteristics of the environment not accounted in the design of the model. In this particularly simplified case the most repeated failures is that the robot arrives to a local minimum (dead end in the straight trajectory between two points). Nevertheless other cases of failure have taken place due to a not enough fine parametrization of some of all the tasks (Figure 5).



(a) *ASGUARD* arrives to a *Dead End Corridor*. The Figure in the low right part shows in green where the robot can plan to navigate. The planner cannot find any trajectory towards the goal (yellow dot).

(b) *CREX* falls when walking down from a too high obstacle.

Figure 5: Two cases of failure. The first was caused by limited environment model (5a). The second one due to bad parametrization of the navigation components (5b).

3.3. Proposed Solution

The solution using the Internal Simulator is as follows. First, all the potentially valid connections (paths) are simulated in both directions. This has to be done in both directions because the failure might only occur when going from state i to j but not in the transition from j to i. The Internal Simulator then executes a simulation of the navigation of each path. The simulation is evaluated and a cost for each path $c_{i,j}$ is assigned based on a cost function Φ

$$c_{i,j} = \Phi(\Sigma(i,j)) \mid i \neq j \; and \; i,j \in S \qquad (1)$$

Where S is the set of states or in this case the subgoals, $c_{i,j}$ the cost of going from state i to j. $\Sigma(i,j)$ is the simulation of the execution of the transition from state i to j and Φ, the function that evaluates the cost of the simulated execution. For simplicity, the cost can be set to infinity for the plans which failed in simulation and to 1 for those which succeeded.

$$\Phi = \begin{cases} 1 & \text{if success} \\ \infty & \text{otherwise} \end{cases} \quad (2)$$

Once all the simulations have been performed and evaluated, a *Validation Graph* $G(S, C)$ is built. Where the directed edges are denoted by

$$C = \{c_{i,j}\}\ \forall i, j \mid i \neq j,\ \ i, j \in S\ and\ c_{i,j} \neq \infty \quad (3)$$

and the vertices correspond with S the states.

This graph can be used to find a validated solution using a graph search algorithm. The solution is then executed.

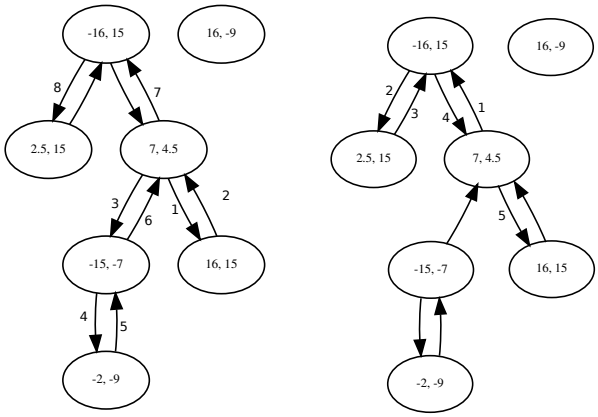
In the generated Validation Graph, the nodes represent the different initial and final points of a path (i.e. initial and final states of the proposed plan). Each directed edge encodes the cost of executing that particular path. The resulting graphs after removing the edges with infinite cost are directed graphs. In particular the graphs may contain parts connected only with only one edge (i.e. either leaving or arriving to them is impossible) and unconnected components (i.e. unaccessible from other components)

In Figure 6 the graphs resulting from running the experiments with both robots are presented. In both graphs there is an unconnected node $(16, 9)$. This is because no execution was able to neither reach the subgoal from another subgoal nor get from that node to any other. Assuming that the start position is $(7, 4.5)$, *Asguard* for robot it is possible to visit all the subgoals with exception of the one that correspond to the unconnected node. In the case of *Crex*, two more states (i.e. subgoals) are unreachable from the initial position. Those states are only connected through an incoming path to the symmetric component where the initial state is.

A comparison of the system with and without the Internal Simulator would show how, in this experiment the robot would enter a failure state. One of the goals (16, -9) is neither reachable from any of the other goal nor the initial position. Thus, a global planner which assumes that every two goals are reachable will fail in this specific mission (independently of the policy of search). Furthermore, the failure could be critical because some of the paths proposed will likely make the robot fall. The solution using the Internal Simulator will neither reach to all the goals, but those paths which were found to generate a failure in simulation won't be executed increasing the chances of reaching more goals.

4. CONCLUSIONS

In space robotics robustness while performing autonomous missions is a crucial and hard problem. The systems are deployed in an environment where they are



(a) Validated paths for the wheeled robot *ASGUARD*.

(b) Validated Paths for the legged robot *CREX*.

Figure 6: The Internal Simulator executes the plan generated between every two subgoals in both directions to find out with paths are valid. The results are stored in a directed graph. The numbers on the edges indicate validated solutions for the mission.

expected to work for long periods of time without maintenance. Ideally, the system would adapt its software to the current physical reality and overcome previously unknown design limitations. Autonomous adaptations, unknown environments and changes in the physical robot are potential sources of failures at execution time in space missions. Validation of the system behavior in an Onboard simulation before execution is proposed to minimize these problems.

The Internal Simulator will operate in close relation with the planners, the environment representation and robot representation. Using this information to pursue to predict where failures might occur. The challenges for an effective Internal Simulator can be summarized into generative (i.e. realistic simulations production), evaluative (i.e. performance supervision), executive (i.e. computational and time constraints) and adaptive (i.e. integrating knowledge from the simulation). The Internal Simulator is conceptually general enough to work with different planners and in different time ranges (*Prediction Horizons* in [RmGA$^+$14]).

A simulation of an exploration mission is presented as conceptual experiment to show the potential of the Internal Simulator. The robot must reach a set of goal points given its 2D position in a non visited area. The system has a serious limitation when attempting to execute the mission: no global map is built. The global map only contains the positions of the goals and the current position of the robot and only the robot position updates on execution. This map is used to generate straight trajectories towards the subgoals under the assumption that they will be traversable. This assumption is often wrong and the robot fails in execution time.

Assuming available a post processed aerial image of the

surface, it is presented how an Internal Simulator would improve the behavior of the system. In the proposed solution, the final path through as many as possible subgoals has to be obtained from a directed graph, which encodes which sub plans have been validated by the Internal Simulator (*Validation Graph*).

This structure represents known states for the Internal Simulator of the planner and whether its transition is actually prune to fail. The structure is then analyzed, in this particular case to go generate a plan that passes through as many nodes as possible as efficiently as possible. This same structure could also be used for different tasks (e.g. get to certain position as fast as possible). Furthermore, it is appropriate to be used with any state based planner

The experiment aims to emphasize that an Internal Simulator to improve the behavior of an state based planner without modifying it by exploiting information of the environment representation not accounted, analyzing the executive features of the whole system (e.g. detect that some transitions might fail due to inaccurate parametrization of subtasks)

In future works it is intended to improve the generative capabilities of the Internal Simulator so that realistic simulations are generated automatically from the environmental knowledge available and using the experiences of the robots. The Internal Simulator will be tested with different real robots in scenarios that resemble challenges of extraterrestrial missions (e.g. lava tubes). It is also envisioned the use with other types of planners (e.g. manipulation) or even combinations of various (e.g. navigate and grasp).

It is envisioned a future were robots will be able to test themselves and discover new strategies and behaviors while reassuring the success of the missions.

ACKNOWLEDGMENTS

The Entern project is funded by the Space Agency of the German Aerospace Center with federal funds of the Federal Ministry of Economics and Technology (BMWi) in accordance with the parliamentary resolution of the German Parliament, grant no. 50RA1407.

REFERENCES

[Bar99] Lawrence W Barsalou. Perceptual symbol systems. *Behavioral and Brain Sciences*, 22(4):577–609; discussion 610–660, August 1999.

[CM09] Antonio Chella and Irene Macaluso. The perception loop in CiceRobot, a museum guide robot. *Neurocomputing*, 72(4-6):760–766, January 2009.

[Hes12] Germund Hesslow. The current status of the simulation theory of cognition. *Brain research*, 1428:71–9, January 2012.

[KRS+12] Tim Köhler, Christian Rauch, Martin Schröer, Elmar Berghöfer, and Frank Kirchner. Concept of a Biologically Inspired Robust Behaviour Control System. In *Proceedings of International Conference on Intelligent Robotics and Applications 2012*, volume 7507, pages 486–495. Springer Berlin Heidelberg, 2012.

[Mar] Mars. An Open-Source, flexible 3D physical simulation framework. http://rock-simulation.github.io/mars/, Last visited: 2015-30-04.

[Mel88] Barlett W. Mel. MURPHY: A robot that learns by doing. *Neural information processing systems*, pages 544–553, 1988.

[MH09] Hugo Gravato Marques and Owen Holland. Architectures for functional imagination. *Neurocomputing*, 72(4-6):743–759, January 2009.

[ODE] ODE. Open Dynamics Engine. http://www.ode.org/, Last visited: 2015-30-04.

[RmGA+14] Jürgen Roß mann, Eric Guiffo Kaigom, Linus Atorf, Malte Rast, Georgij Grinshpun, and Christian Schlette. Mental Models for Intelligent Systems: eRobotics Enables New Approaches to Simulation-Based AI. *KI - Künstliche Intelligenz*, 28(2):101–110, March 2014.

[Roc] Rock. The Robot Contruction Kit. http://rock-robotics.org, Last visited: 2015-30-04.

[SB12] Richard S Sutton and Andrew G Barto. *Reinforcement Learning : An Introduction (Second Edition)*. 2012. Chapter 8: Planning and Learning with Tabular Methods.

[VZ06] Richard Vaughan and Mauricio Zuluaga. Use your illusion: Sensorimotor self-simulation allows complex agents to plan with incomplete self-knowledge. *From Animals to Animats 9*, 4095:298–309, 2006.