592541

A new simulation framework for autonomy in robotic missions

Lorenzo Flückiger, Christian Neukom

NASA Ames Research Center - QSS Group Inc.

Abstract

Autonomy is a key factor in remote robotic exploration and there is significant activity addressing the application of autonomy to remote robots. It has become increasingly important to have simulation tools available to test the autonomy algorithms. While industrial robotics benefits from a variety of high quality simulation tools, researchers developing autonomous software are still dependent primarily on block-world simulations. The Mission Simulation Facility (MSF) project addresses this shortcoming with a simulation toolkit that will enable developers of autonomous control systems to test their system's performance against a set of integrated, standardized simulations of NASA mission scenarios. MSF provides a distributed architecture that connects the autonomous system to a set of simulated components replacing the robot hardware and its environment.

1 Introduction

Exploration of remote environments using robotic systems is a very challenging task. The physical robot has not only to withstand the hostile elements of the environment, it also needs to demonstrate high levels of autonomy to accomplish its tasks without continuous human control or intervention.

The typical human-robot interaction scenario for an exploration rover on Mars is strongly affected by the communication issue: scientists on earth can only communicate with the Martian rover once a day, with a limited bandwidth and for a period of one to two hours. Communication becomes even a greater issue for planets farther away from earth like Europa. On Earth, the exploration of some remote places like the ocean under the Arctic ice faces similar problems. Therefore, besides the development of ground tools to help scientists plan, test and visualize remote science experiments, the rover needs autonomy to perform tasks without direct human control. To accomplish critical science missions, the rover needs to be able to explore an unstructured world and to make decisions based on on-board sensors. In addition, the rover has to be capable of adapting its mission to unanticipated events in order to maximize the science return and ensure rover safety.

Research in autonomy for robotic platforms used in remote exploration is therefore critical for the success of missions. However, this research area faces problems when it comes to testing new autonomy concepts:

- Custom simulators provide usually only testbeds for individual algorithms without the context of an integrated robotic mission.
- Access to hardware is either very costly or not available at all and even if the hard ware is available, time and resource constraints limit test scenarios to just a few environments. In addition, it is difficult to control the parameters of the test and the experiments are therefore hard to repeat.

To support autonomy research for robotic systems, the Mission Facility (MSF) project was started in 2001.

1.1 MSF Goals

The goal of MSF is to develop a simulation framework and suite of simulation tools to support research in autonomy for remote exploration. Such a system will allow developers of autonomous software to test their models in a high-fidelity simulation and evaluate their system's performance against a set of integrated, standardized simulations. Currently there is a big gap between autonomy software at the research level and software that is ready for mission insertion. It is our vision that MSF will bridge this gap by providing researchers with high-fidelity simulations of mission scenarios to test their software in a realistic, complex environment.

Software simulations have become very common and many high fidelity models exist in the robotics community. However, it is currently very difficult to combine models of different origin into an integrated simulation because the various models may be tied to a specific operating system or computer language. MSF is designed to connect these models through the Mission Simulation Toolkit (MSTK), a soft-

ware package comprising 1) a framework for connecting and synchronizing distributed software models, 2) generic interfaces abstracted from the transport layer, and 3) a set of basic components needed for a simulation.

1.2 Scope and Applications

The MSF architecture is designed to support multiple mission platforms (e.g. planetary robots, spacecraft, underwater vehicles), but the initial focus is on supporting simulations in the domain of planetary rovers. A rover simulation would most likely include a terrain model, site information (such as coloration and mineral composition), an environment model (sun position, lighting, temperature, etc), a rover including a kinematic model and on-board sensors, and some scientific instruments. It is up to the user to decide what granularity of models best suits the purpose of his simulation. For example, a user who is mainly concerned with collecting scientific data may not require a sophisticated rover model because he may not care how the rover gets from one point of interest to another. On the other hand, a user who is developing a trajectory generator may want to control individual wheels of a rover. MSF provides both high and low level interfaces to a number of standard models. Figure 1 shows the concepts of MSF distributed simulation relying on a common communication framework to connect models and autonomous software.

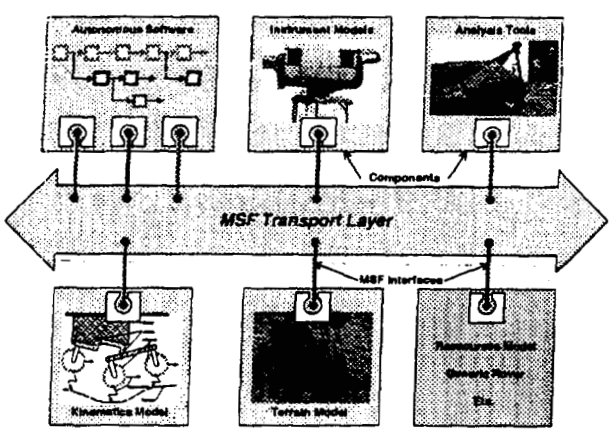

Figure 1: Overview of the Mission Simulation Facility (MSF).

2 Proposed Framework

The realization of the goals mentioned in the previous section requires the construction of the following components:

A distributed architecture that implements a communication layer.

A set of models allowing the simulation of robots, environment and science instruments.

Scenario representation through the description of robot systems and their environment.

Databases used both as a source of data for the model and as storage for the result of the simulation.

These components (except databases which are not yet treated by the MSF team) are described in the following sections.

2.1 Distributed architecture

The MSF architecture is derived from two main requirements: to support distributed simulation on multiple platforms and to ensure extensibility through an open architecture.

Multiple platform support. The users of the MSF (autonomy developers) typically develop their tools in a variety of environments, for example, a Lisp program under Solaris on a Sun workstation or a C++ program under Linux on an Intel PC. In addition, the target systems for the autonomous software (the rover control software) are also developed on different operating systems and hardware platforms. Such software could for example rely on a particular flavor of Linux running on a PC-104 Pentium board, or could be a dedicated embedded system running VxWorks. The MSF project does not intend to develop all the simulation components but rather will take advantage of existing tools. To minimize the adaptation, each particular software component should be usable by MSF on the original platform for which it was developed.

Open architecture. MSF is a general purpose testbed for mission simulation rather than a specific simulator, which implies that different sets of components will be used for different scenarios or different domains. For example, one might use a kinematics model for a rover and a fluid dynamics model for an underwater vehicle. In addition, a component should be usable in multiple scenarios, which means that the same rover kinematics model should be usable for various rovers with particular payloads driving in different environments. Finally, it should be possible to replace a component of a certain type with another that performs the same function but at a different level of fidelity. A simple kinematics rover simulator could be sufficient to test high level autonomy concepts like path planning while a dynamics rover simulator including accurate soil-wheel interactions may be required to test an autonomous control

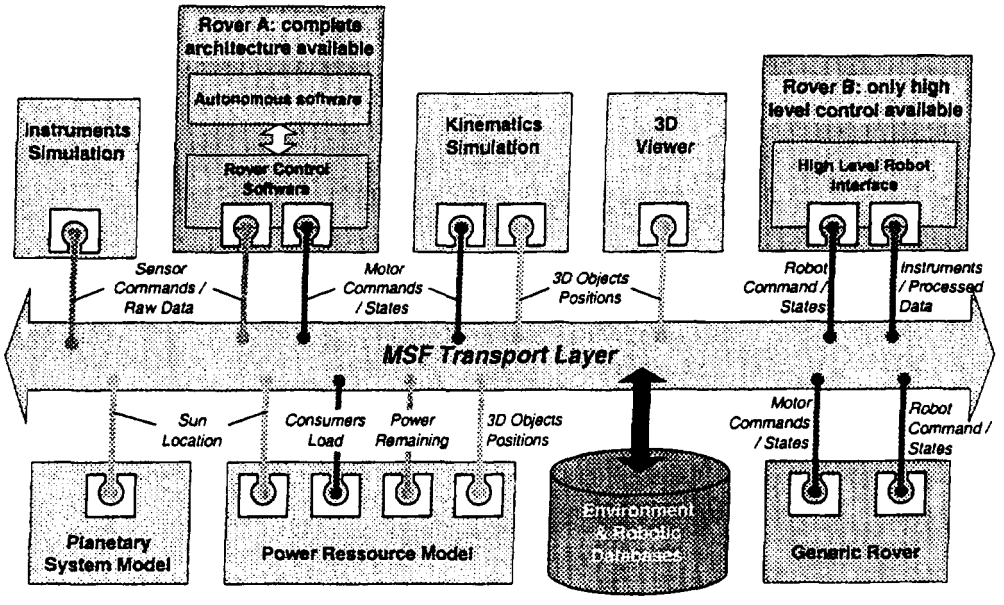

Figure 2: MSF simulation with several components communicating through common interfaces.

system for the mobility of the rover[1]. It is therefore essential that MSF define clear interfaces between the components to facilitate swapping components included in the MSF toolkit with those developed at other research institutions.

A way to satisfy the above requirements is to have a distributed architecture where components communicate with each other using a common transport layer. MSF is built on top of the standardized High Level Architecture (HLA)[2] which is an architecture for simulation reuse and inter-operability developed by the Defense Modeling and Simulation Office (DMSO). MSF currently uses the Runtime Infrastructure (RTI) – a software implementation of HLA – freely distributed by DMSO. The HLA/RTI provides MSF the following services:

- Multi-platform support: IRIX, Solaris, Linux, Win32 and VxWorks
- C++ and Java bindings
- Choice of transport protocol: TCP (reliable) or UDP (fast)
- Publish/Subscribe scheme
- Communication through objects or messages
- Various time management schemes for simulation synchronization

To facilitate the integration of components in an MSF based simulation, an abstraction layer has been developed on top of the HLA, the Federate ToolKit (FTK). FTK is responsible for the integration of communication entities with the Runtime Infrastructure. The communication objects and messages defining the MSF interface are easily designed using the Unified Modeling Language (UML) and all the necessary C++ code to use these communication entities can be automatically generated.

Figure 2 gives an example of an MSF simulation with several interconnected components. Two separate rover autonomy software executions are participating in the simulation: Rover A software is provided by a lab having a complete rover architecture (and probably a real rover) from the high level control down to the hardware control; Rover B software comes from a lab working only on high level autonomous algorithms and does not have hardware control. When Rover A software is sending commands to its actuators (e.g. motor1.start(speed, duration)), the commands are routed to the simulator rather than going to real hardware. The Kinematics Simulator accepts such commands and computes the behavior of the rover on the terrain regarding these inputs. When Rover B is issuing a high level command to its base controller (e.g. roverB.moveto(position, obstacle-avoidance=on)), the Generic Rover model catches this message and produces motor commands for a simple model of a rover causing it to move from one location to another while avoiding obstacles. These motor commands can be processed by the same Kinematics Simulator that is used for controlling Rover A. The same type of scheme is used when controlling instruments, generalized as sensors in Figure 2 (the figure does not show the full path

[1]This latter case is however not directly addressed by MSF in its first phase

[2]The HLA was approved as an open standard through the Institute of Electrical and Electronic Engineers (IEEE) - IEEE Standard 1516 - in September 2000

of information flow). In addition to the Kinematics and Instruments models, a Power Resource model is participating in the simulation. Its function is to monitor the load of each actuator as well as the power generated by solar panels and to compute the power remaining in the rover batteries. The power output of the solar panels depends on the orientation of the solar panels relative to the sun. The position of the panels is provided by the kinematics model and the sun's position is delivered by a another component computing the Ephemerides. This example shows how different components are reusable for different scenarios, and how the definitions of the networked MSF interfaces removes all the dependencies between the components.

2.2 Simulation Models

MSF makes it possible to replace robotic hardware and environments with simulated components, which implies having a library of models of the robots themselves and of various environments representative of future missions. The interaction of a rover model with its environment involves several elements. For example, when the robot moves to a new location, the motors drive the wheels, which in turn move over the terrain according to the forces of gravity and terrain shape. The robot obtains engineering data through its sensors, and on-board science instruments collect data from the virtual environment. Robotic instruments such as a rock grinder may also interact with the environment.

The models for a simulation could be built at very different levels of sophistication. For example the model of a rover driving over some terrain could work with the assumption of a flat surface and perfect motion, or could involve a full dynamic system including wheel slippage. The next paragraph briefly presents different approaches.

Level of simulation. The purpose of the science robotic system is to explore its surroundings by actively taking measurements. Figure 3 depicts a number of modules composing a rover software architecture with flows of two types of information: action (output) and sensing (input). In this simplified example, the Goal Decomposer on the Rover Autonomy side hands tasks to a Path Planner, which in turn interacts with a Locomotion Controller. The latter is communicating with the rover hardware, sending commands to actuators and obtaining proprioceptive sensor data. On the side labeled "Science Autonomy" in the figure, the world is sensed using a camera controlled by a Vision System that feeds an Image Interpreter module. The latter supplies information to high level World Analysis algorithms. In real autonomous systems, the flow of commands and sensory inputs are closely connected. For example, the obstacle avoidance algorithm takes actions based on the information provided by the obstacle detection algorithm. While there is a fair amount of cross-over of information between the science and rover autonomy, generally more sensory input data is directed to the science autonomy side and much of the output is concerned with the functioning of the Rover Autonomy. The simulator could be hooked at any of these levels of the rover architecture on each side, and not necessarily at the same level for each area of specialization.

It is interesting to note that in general it is easier to build a simulator that connects at the low level of the "action flow" and at the high level of the "sensing flow". On the "action flow" side, building a simulator that sends low level commands to the actuators is easier because: 1) access to hardware is less dependent of the rover software architecture and 2) the simulator does not need to replace some behaviors of the rover (e.g. obstacle avoidance in the example above). In contrast, a simulator could more easily feed the high level modules of "sensing flow because": 1) the simulator knows the truth model and thus can directly provide high level information about the scene (for example, there is a big rock in front of the rover) and 2) an accurate instrument model that takes into account physical laws (illumination, material texture, etc) is difficult to build.

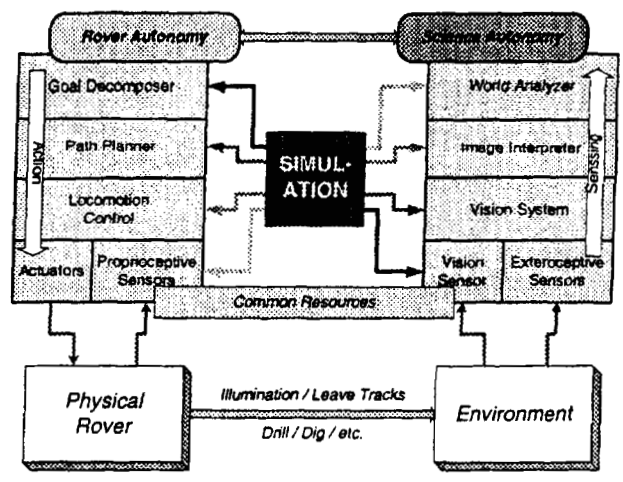

Figure 3: Various level of simulation possible for a hierarchical robot software architecture.

MSF is being built incrementally, so our initial focus is on creating a strong infrastructure and providing a toolkit that includes only a few simple models. Once the infrastructure is in place, new models of higher fidelity or models that address different domains will be added. Currently MSF depends on an existing kinematics simulator for the rover mechan-

ics (controlled by low level inputs like wheel speed) and feeds control algorithms with high level information coming from the truth model (like "this rock contains carbonates"). Section 3 briefly presents the components used in the current MSF implementation. Ongoing collaborations will provide improved models for terrain generation, dynamics simulation and science instruments.

Required Components. MSF is a general framework for connecting multiple components to build various types of simulations. In this sense, MSF does not enforce the use of a particular set of simulators, and lets the user build to meet his own needs. However, it is possible to identify the components required for a minimal planetary simulation:

Environment Generation. An environment model is essentially composed of a terrain surface with attached properties. The data can either be generated on the fly by a terrain server, or precomputed and stored in a database. Ideally the data will be generated by terrain models from a set of input parameters (location, resolution, type of terrain, rock density, etc.) but real terrain data can be used as well in certain situations (MSF currently uses data collected during various Mars mission and Earth field tests). An environment model may also comprise data representing atmospheric conditions, celestial body positions, and soil composition, etc.

Robot Behavior. The robot model represents mainly the mechanical behavior (positions of robot parts in response to commands to effectors) because it is the most significant from a high level point of view.

From a high level point of view, the most significant representation of the rover model is the mechanical behavior, which determines the positions of robot and its effectors. The robot model also needs to include other resources including proprioceptive sensors, power, computation and communication. Depending on the level of autonomy being tested, the model may also include generic robot capabilities such as obstacle avoidance. Finally, it is critical for the robot simulation to represent and handle failures because they play an important role in robot autonomy.

Instrument Models. Instrument models provide autonomy developers data from the robot's environment, feeding their algorithms. Again, depending on the level of autonomy being tested, these data products could be the raw data an instrument normally returns (e.g. a photo-realistic image) or pre-processed information (e.g. the response from a spectrometer system: "there is carbonate in this rock"). MSF will provide models for generic instruments including cameras and spectrometers. Complete instrument packages as those intended to fly on future mission may be added to the library as the models become available.

Data Analysis. Tools for collecting and analyzing data from simulation runs are fundamental to evaluate autonomous software. The HLA communication layer and the tools developed for it make it easy to collect all information exchanged between MSF components. One of the first qualitative evaluation tool used in MSF is a 3D graphic environment allowing the user to follow robot behaviors in a simulated world. Beside showing graphics objects having a physical counterpart, the 3D environment provide visualization on non visible parameters like the torque on a motor, or the field of view of a camera.

2.3 Scenario description

Because MSF is a general simulation platform rather than a specific simulator, it needs to be configurable for different scenarios. In general, a scenario comprises of the robotic systems description, the environment description, and some simulation specific parameters. The robot description defines the kinematics (or dynamics) parameters of the robot's mechanical structure, the actuators, various instruments composing the payload[3] and several resource modules (e.g. battery, memory). The environment is composed of the following: 1) a reference to the terrain data that will be used to compute the kinematics of the rover, 2) the data inferred by the instruments, 3) the type of atmosphere and other environmental parameters, 4) the location of the mission (which planet, latitude and longitude) and 5) the epoch of the simulation.

No parameters regarding robots or environment are encoded in any MSF component. Instead, the components obtain their specifications from the scenario description file when a new simulation is initiated. A component defined in such a way will then produce simulated results on the basis of an external description. The scenario description file could in fact be a set of files, each referenced in a main file. Such a setup creates a unique parameter, the file URL, defining completely a specific scenario in a MSF simulation. MSF currently uses a file description that is a proprietary extension of the Robot File Format

[3]Proprioceptive sensors, such as the incremental counter of a DC motor, are considered part of the actuator. Low level control between a DC motor and its controller is not simulated at the level of accuracy that MSF addresses.

used by VirtualRobot, which does not contain all the capabilities listed above. MSF plans to use an XML scenario description with a dedicated DTD. This will provide a description of the simulation that is human readable (and editable) with the bonus of having existing graphical editors to modify the file and the necessary tools to parse it. In addition, tools to exchange XML files over the network are widely available, which is important for a distributed simulation.

To ensure coherency, all the MSF components participating in a simulation read the same scenario description file. However, each component extracts only the information relevant to its domain. For example, the kinematics simulator identifies all the dimensional parameters of a rover, while the instrument model only needs to get the rover's name. Having a single source for describing the scenario is critical for software configuration management and testing. A seemingly simple change to the model robot such as increasing the power of the motors, for example, affects multiple software components: the power model, dynamic simulator, abstract reasoning layer, and possibly the visual representation, to name a few. Without limiting the scenario description to one file, the resulting simulation could become unmanageable to verify and distribute.

3 Results: a first MSF instantiation with three components

3.1 Conditional exec

3.2 Kinematic simulation

3.3 Visualization

4 Conclusion and Future developments

In this paper, MSF has been presented as a simulation framework for autonomy research that makes it possible to integrate available models of robotics systems with autonomy software. The distributed architecture that implements the communication layer enables researchers to run their models on the intended platforms. The toolkit provides users a facile method to create communication entities, that connect their software with other models participating in a simulation, through specification in UML. A library of simple and advanced models of robots, science instruments and environment allows users to create a simulation for a particular scenario. It is our hope that this library will grow as more and more users will use MSF and make their models available to other research groups. We are also hopeful that the flexible design and ease of use will make MSF attractive to a large community of users interested in creating simulations in robotics and autonomy.