A High Level Teleoperation Platform for Space Robotic Missions

Marco Chacin, Eric Rohmer, Andres Mora and Kazuya Yoshida
Department of Aerospace Engineering, Tohoku University
Aoba 6-6-01, Sendai, 980-8579, JAPAN
{mchacin, eric, andresmora, yoshida}@astro.mech.tohoku.ac.jp

Abstract

This paper describes the ongoing development of a teleoperation platform for space robotic missions. The proposed platform consists of a ground control station, an offline simulation environment, a full-scale working model of the robot and the actual robot on the mission. From the simulation environment a path strategy is obtained and a set of commands is generated and sent to the ground control station using standardized files (XML) to describe how commands and data move between functional units, this way it is no longer needed to create custom software, and/or distributed objects to communicate with new systems. Instead, only providing an XML description of how the communication takes place will be required. Then the systems can parse and interpret the XML to generate the required commands to control the unit, interface with it, construct messages and display the data that is returned. A software prototype is developed for an asteroid exploration mission to evaluate the proposed platform.

1. Introduction

The growing interest in space exploration missions due to the success of missions like Spirit and Opportunity [2], has led to an increased the number of missions teleoperated from Earth. In order to keep the pace of productivity gains are necessary; for this purpose reusability of components needs to be considered.

In parallel to this growing interest, cooperation between space agencies has increased as well. This implies that different platforms need to communicate with each other, making interoperability among such platforms an issue.

The Space Robotics Laboratory at Tohoku University is facing the same reusability and interoperability issues in order to avoid redundancy in development and implementation phases.

In this paper, the authors are introducing a novel standardized way to communicate among platforms, leading to the reusability of resources. Based on a dialect of XML, commands and data from different platforms can be exchanged and also extended to new modules. With this new approach, the information traded will be suitable for cross-platform applications and manipulation with relatively simple software or directly by a human.

Following a short description of the Japanese Experimental Module Remote Manipulator System (JEMRMS) platform developed and currently used at the Space Robotics Laboratory [1], the authors present an overview of the high level teleoperation platform for space robotic missions that deal with the reusability and interoperability issues. Then the proposed standardized communication specification called Space Teleoperation Markup Language (STML) is detailed, and applied to an asteroid surface exploration robot mission as an example.

2. The JEMRMS simulator

The JEMRMS is a macro-micro system to be mounted on the International Space Station (ISS). This module consists of two manipulator arms called the Main Arm and the Small Fine Arm. In recent years, the Space Robotics Laboratory has developed a platform to aid maneuver the JEMRMS [3]. This platform has a real-time motion simulator that uses a dynamics engine. It is also capable of rendering in real-time 3D images of the JEMRMS from three different viewpoints, so that a human operator can observe the dynamic response of the arm reacting to the motion commands given through joysticks. By the use of this platform the vibrational behavior of the Main Arm excited by the reaction of the Small Fine Arm can be studied.

The general configuration of the platform can be seen in Fig.1. The simulation environment and the operation environment are connected by sockets transmitting the data back and forward between the two environments through an Ethernet link. The operation environment consists of one display for the GUI of the system and three other monitors for the 3D graphics of the JEMRMS, the computer of this environment runs under Windows XP. Inputs to the system can be done by the keyboard, or two different joysticks. The GUI at the operation environment is written in Java code and

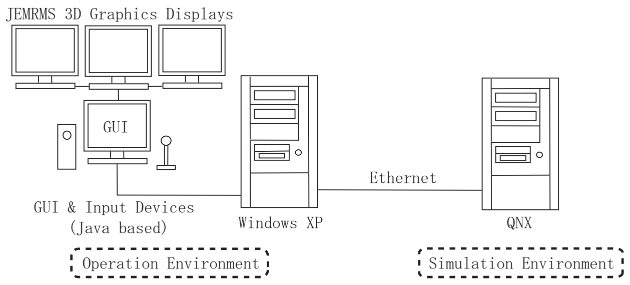


Figure 1. Configuration of the platform.

presents the 3D model of the JEMRMS using a 3D CAD that is loaded on the GUI through the Java 3D API.

In Fig.2 the actual platform with an operator is shown. In the monitors displaying the 3D model of the JEMRMS, the operator first moves a "ghost" image of the manipulator, checking that the path taken will not collide with obstacles, taking torque measurements, etc. Then, when the operation is acknowledged the commands describing the actions of the simulation are sent to the actual manipulator (in this case the operation environment), which can be visualized at the monitors following the ghost's path.

Once the simulation environment has received the data from the operation environment it will be used to calculate the dynamics of the movement the operator desires to perform. This is accomplished in real-time, using a computer running under QNX.

This platform has shown itself useful as a tool to determine path strategies at the JEMRMS, but its capacity to be reused for other applications depends on major software and hardware changes. Indeed, if another model is desired to be implemented rather than the one which the platform was designed for, the dynamics model running in the real-time operating system has to be changed as well as its graphical representation on Java 3D (running under Windows XP).

If the platform is desired to be extended, the data exchange among the distinct environments should be designed from scratch, instead of reusing a standardized file or stream of data with the new added module or section of the platform.

3. Overview of the platform

The platform proposed will tackle the issues presented in the previous section using standardized format files. These files will be shared by the various modules of the platform so that a parser at each module will determine what part of the file has the information or commands it requires.

The main modules of the platform are: the mission, the ground control, the simulation, the decision and the information access module as presented in Fig.3. The following describes the purpose of each of the modules of the high

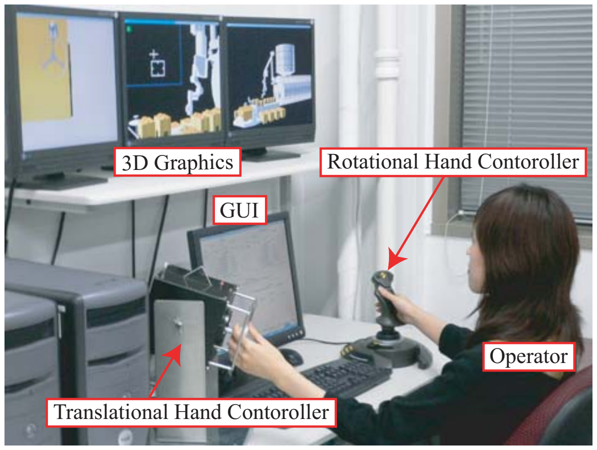


Figure 2. Operator at the JEMRMS simulator.

level teleoperation platform for space robotic missions proposed.

3.1. Mission module

The authors consider a "mission" as any remotely controlled robot on a foreign planet and/or outer space from a ground station on the Earth. The robot will be able to take information from its surroundings, enabling the operators on the Earth create 3D maps and making navigation feasible. Independently of the type of mission, the basic structure platform will not change.

The mission will wait for a set of commands and will send back to Earth sensory and scientific information gathered at the site.

3.2. Ground station module

The ground control station is the link between the Earth and the mission. The communication interface is able to manage the connection with the mission, uploading the files that contain the sequences of commands which the mission will follow and also downloading files that contain the sensory information,internal sensors for odometry, external sensors to define the environment and the data from scientific instruments.

The control panel is able to parse the STML file containing the set of commands to be sent to the mission. It can also remotely trigger some programmed behaviors of the robot in the mission such as reboots, turn on/off the actuators in order to save the battery or start an automatic sequence. It encodes the information coming from the mission into STML that will be sent to the simulation module and makes the scientific and public data available to the information access module.

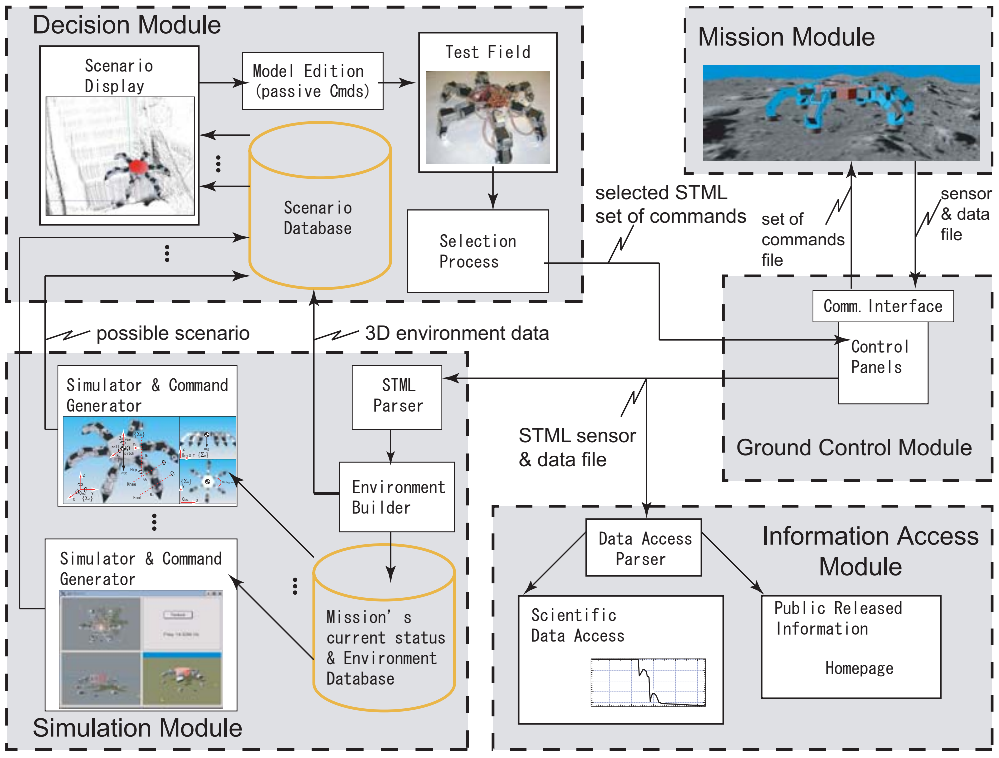


Figure 3. Overview of the High Level Teleoperation Platform for Space Robotic Missions.

3.3. Simulation module

The purpose of this module is to simulate and generate the possible scenarios in an STML file. It is composed by an STML parser that extracts the sensory and odometry data provided by the ground control module that represent the latest status and environment of the mission.

Once the information has been extracted from the file, the environment builder generates the database of the refined status and environment of the remote robot for the simulator and command generators. At this point, several operators can share the database to run their own simulation and generate scenarios to be studied.

Using an external database file any simulation environment can generate a scenario with minor implementation efforts. It can be noticed that the parser is in a separate submodule because not all the simulation environments can understand XML tags but the authors assume that all of them can read and generate files. Once an appropriate scenario has been built, the operator will send it to the decision module.

3.4. Decision module

In this module the data of the mission is analyzed to define strategies for the next step of the mission. The selection process is done so that the STML command files coming from the simulation module can be replayed and edited to add some passive commands for the robot (i.e. taking pictures, scanning the environment, any instruction that would not require any re-run of the simulation, etc).

To help decision makers select the appropriate list of commands, the scenarios can be replayed in a virtual 3D environment and in a real test field with a mock-up of the actual robot used in the mission. The environment of the robot in the test field will be manually reconstructed as it is done for the Spirit and Opportunity missions at the Mars Yard.

Once the mission deputy manager decides which strategy should be used, the upgraded set of commands file is forwarded to the ground control module for its upload to the mission.

3.5. Information access module

The information access module receives the catalogued data from the ground control module through the data access parser.

This data could be released either to the public through automatically generated HTML documents or to scientists in raw formats or as graphs.

4. XML for space missions

There have appeared several proposed common languages for communication between systems and while some of them are powerful and convenient for expressing the information related to their respective applications, the simultaneous utilization of a robot programming language and a communication language in a single application creates significant difficulty for the human operator.

Thus, the goal is to propose a language specification that would unify the languages of the three categories considered above. Namely, the specification should:

- Be powerful enough to contain everything that can be expressed in the mentioned languages for robotic programming and communication;
- Allow its manipulation via simple software tools or directly by a human;
- Be suitable for cross-platform systems and applications that must be easy to develop, modify, maintain, extend and reuse utilizing emerging technologies while maintaining the legacy between older versions of the specification;
- Reduce the implementation time used by scientists and engineers by defining the interface between hardware and software engineers; and
- Separate implementation from description.

In order to satisfy these requirements we have chosen to use Extensible Markup Language (XML) as a base to develop the specification of Space Teleoperation Markup Language (STML), a markup language for Space Robotic Missions. STML is being developed with the goal of creating a very general and highly extensible framework that applies to any kind of mission that can be controlled from Earth.

In short, STML is a specification that uses XML to aid in the exchange of data and commands between modules. Furthermore, XML has proven to be convenient to describe various types of structure data, thus the availability of the specifications of XML-based languages for many engineering and scientific domains [4] makes it possible to utilize their syntactic models and to reduce the work on the design of a new language.

Although XML itself is not sent between systems, XML can be used to describe the commands and data that can be transferred. For instance, since XML-compliant code is easy to parse and generate by current generation programming languages, any software tool could read the files and interpret the incoming data. It is this ability to manipulate the language code with the help of relatively simple software and its transparency to the user that can be particularly important for maintenance.

Ultimately, by using XML to describe how commands and data move between computers and modules, it will no longer be needed to create custom software and/or distributed objects to communicate with different platforms. Instead, future mission software will only need to provide an XML description of how the communication takes place. Tools that can parse and interpret the XML will be able to automatically display the data that is returned from the different modules due to the fact that parsers, translators and other types of software components for XML are available for many applications and most computer platforms.

STML represents a first attempt at making this capability a reality.

5. The STML specification

In this section the main concepts of STML are outlined. For the sake of clarity the language constructs that are not related to the vocabulary of STML are omitted.

A valid STML document will begin with a MISSION tag. This tag must contain one or more robot tags nested to any arbitrary level where nested tags represent subsystems. Intermingled with nested containers can be an arbitrary number of Commands and a general description of the robot morphology. STML, therefore, has a basic structure like:

```
<mission>
     <robot>
        <structure>
        </structure>
        <commands>
        </commands>
     </robot>
</mission>
```

The MISSION tag also includes several attributes used to identify the mission itself. The attribute list starts with ID, which states the assigned identifier for the mission, the mission launch date, the name of the mission, a short description of the mission, its status and the version of the specification used for the management of the mission.

Next, the robot's hardware can usually be represented as a semantic tree according to the physical and logical interconnection of the units [5]. This representation is particularly suitable for the nested markup structure of XML-based languages. This fact is exploited through the utilization of the container elements. These tags content an abstract representation of the robot and its functional part as a dynamic body.

The STRUCTURE tag Fig.4 has two attributes, the first one describes the robot and the hidden attribute is Boolean and determines if the structure of the robot is to be shown in the user interface.

```
<structure description="" hidden="">
    <body id="" frame="" inertiaParam="">
        <link id="" joint="" axis="" frame="" inertiaParam="">
            <childLink id="" joint="" axis="" frame="" inertiaParam="" />
            <childLink id="" joint="" axis="" frame="" inertiaParam="" />
        </link>
        <link id="" joint="" axis="" frame="" inertiaParam="">
            <childLink id="" joint="" axis="" frame="" inertiaParam="" />
            <childLink id="" joint="" axis="" frame="" inertiaParam="" />
        </link>
    </body>
</structure>
```

Figure 4. STML description of the robot structure.

```
<commands>
    <command name="">
        <argument name="" datatype="" required="" hidden="" timestamp="" />
        <argument name="" datatype="" required="" hidden="" timestamp="" />
    </command>
    <telemetry name="">
        <sensor name="" datatype="" required="" hidden="" timestamp="" />
        <sensor name="" datatype="" required="" hidden="" timestamp="" />
    </telemetry>
</commands>
```

Figure 5. Basic STML description of a command sequence.

Going down yet another level, a referential frame is attached to the BODY tag; this represents the main body of the whole mechanical structure of the robot, which also possesses an ID attribute and its correspondent inertia parameters. Similarly, each LINK tag represent bodies directly connected to the main body, thus two attributes are added. JOINT indicates the type of joint the child link shares with the parent link and AXIS states the axis of translation or rotation of the child link in relation with their parent.

COMMANDS tags Fig.5 are divided into COMMAND and TELEMETRY tags. Commands are to be sent to mission robot while Telemetry contains sensory data from the mission. Commands contain optional arguments and Telemetry will contain one or more fields of data.

Each Command Argument has many attributes. The first required attribute is the Argument's name and the second is the Argument's data type. Currently, STML uses strings to specify a Java data type. The required attribute is optional and specifies if the argument itself is optional. If required is not given as an attribute, then the default value for required is False. As with the Structure tag the hidden attribute determines if the argument appears in the user interface.

Each Telemetry "sensor" also has several attributes; the Sensor's name, the sensor output's data type, hidden attribute and timestamp, many more could be added according to the requirements of the mission.

6. An asteroid exploration mission

The application chosen by the authors to show the efficiency of the platform concerns the next minor body exploration mission, in which a smart design of a robotic system will allow scientists more accurate positioning on the microgravity environment.

With the recent success of the MER Missions [2], there is an increasing interest in robotic exploration to small celestial bodies characterized by a medium to low gravitational environment such as moons, asteroids and comets. Such interest, especially in asteroids and comets, is due to their status as the remnant debris from the inner solar system formation process. As an alternative, hopping systems for planetary exploration were first proposed and used in the MUSES-C mission [7]; using an internal reaction wheel to obtain a thrusting force in order to move on the surface. But even with this idea, the motion of the rover would be hopping and bouncing on the asteroid, therefore the location of the robot when the bounds are finally damped out is very difficult to predict or control.

For the next mission a small rover will be deployed over the surface just before the impact sampling sequence [8] and then it would crawl to the pinpoint location of the fresh crash-hole after the sampling. Further details about the mission's goal can be found in [9]-[10]. We present in this sec-

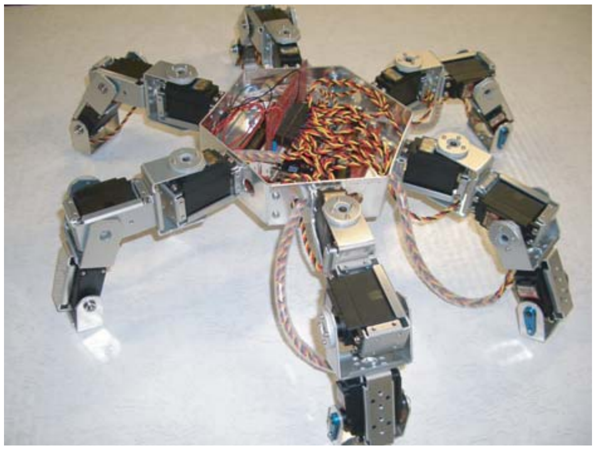
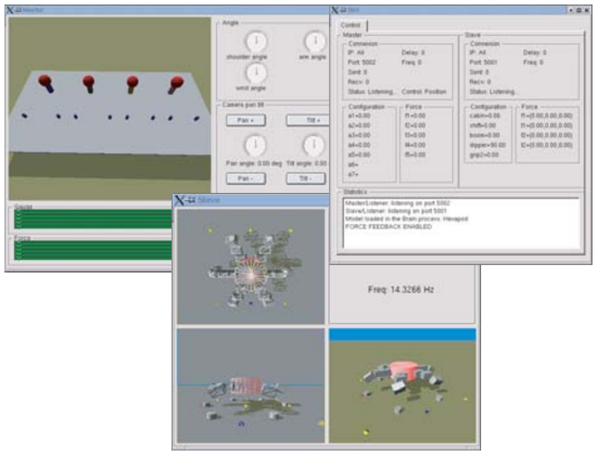


Figure 6. Rover during early stages of development, the electrical systems can be seen.



```
Master/Listener: listening on port 5002
Slave/Listener: listening on port 5001
Model loaded in the Brain process: Hexapod
FORCE FEEDBACK ENABLED
```

Figure 8. Screenshot of the GUI of the path generation simulator.

tion the early stages of the on going development regarding this asteroid exploration mission.

6.1. The robot at the mission

The basic requirement for the robot is to achieve scientific investigation and mapping of the asteroid surface at several locations with fine positioning capability once landed on the asteroid. For this purpose, currently the hexapod [11] is formed by a central body, with a hexagonal shape and six similar legs that sum for a total of 18 DOF, and symmetrically distributed around the body as seen on the early view of the prototype on Fig.6 and as described in STML format in Fig.7.

Each DOF is actuated by a Hitec RC Servomotor as fed by a JStamp+ microcontroller, which uses an aJile processor running at 80MHz with 2MB of RAM.

For these kind of controllers Java is the native instruction set with more than 99% of Java bytecodes microprogrammed in hardware, aJile MPUs deliver Java execution with a memory footprint that is 2 to 3 times more compact than similar 32-bit RISC applications. Among the features are:

- Direct Java Virtual Machine bytecode execution;
- Micro programmed real time Java thread manager;
- Integrated power management; and
- Timers, interrupt controllers, memory controllers and a variety of communication interfaces.

Several types of sensors are to be attached to the controller board, like a Laser Range Finder (LRF) fixed to the body and force sensors on the limbs that will provide the system a model of the local environment of the robot. Data from inclinometers and odometry will be used for orientation and localization as well.

The prototype will be used, at first, as a test bed to simulate the real mission, and to evaluate walking strategies and the full teleoperation platform. The prototype will implement the same behaviors as the robot on the mission and will be used later in the test field during the path selection process. To emulate the gravity of the asteroid the robot could be mounted on the tip of a Mitsubishi PA10 manipulator equipped with a force-torque sensor.

6.2. The simulation environment

A real time dynamic simulation system is being developed based on the previous teleoperation platform of [13]. The aim of this simulation system is to offer the operators an ergonomic path generator that provides a possible set of commands in an STML file for the next motion scenario of the mission. During the development stage, the simulation environment is being used to elaborate and test walking strategies. The simulation environment runs under Windows XP and Linux, uses Open Dynamic Engine for the dynamic simulation and is coded in C++. Fig.8 shows screenshots of the ongoing development simulation with a virtual model of the hexapod.

The latest update of the sensors of the mission is provided in an STML format and parsed to extract from it the sensory data of the LRF and odometry information concerning the orientation and position of the robot. The Elevation Map Builder module could, at this point, process this data using SLAM and referencing algorithms to define a more precise position of the simulated hexapod in its virtual en-

```
<mission id="0001" date="01/01/2012" name="asteroid exploration" description="" status="active" version="1.0">
    <robot type="hexapod">
    <structure description="free-floating - hexagonal shape and six similar limbs" hidden="false">
        <body id="00" frame="world" inertiaParam="">
            <link id="01" joint="rotational" axis="Z" frame="body" inertiaParam="">
                <childLink id="02" joint="rotational" axis="X" frame="parent" inertiaParam="">
                    <childLink id="03" joint="rotational" axis="X" frame="parent" inertiaParam="" />
                </childLink>
            </link>
```

Figure 7. Example STML description of the robot.

vironment and build a refined map of the asteroid. All this information can be used as an entry point for the dynamic simulator. With a joystick and a keyboard, the operator can control in real time the motion of the virtual hexapod while the set of commands is logged in a file using the STML specification. Several operators can work on possible simulation scenarios at the same time. Those files can later be analyzed and discussed in the decision module for the final selection process.

6.3. Control from Earth

As mention before, this section serves as a communications center that connects the mission and the other modules of the teleoperation platform. For this application the authors limited its features to network capabilities. Once the appropriate set of actions is decided by Mission Control, the STML file containing these actions will be parsed in order to extract from it the actual commands to be sent to the robot on the mission. After this stage has been achieved those commands will uploaded to the robot itself.

7. Conclusions and future work

In this paper a standardized communication method to exchange data between different modules inside a robotic platform has been presented.

The concept of a high level teleoperation platform for space robotic missions has been explained and also validated through a minor body exploration mission using a hexapod as the mission's robot.

While the proposed methodologies are being successfully tested, we recognize that much more work needs to be done to refine the formal specifications of the platform and the STML, therefore future challenges of the current research should also include the development of the environment builder at the simulation module as well as a test field where experiments can be done.

References

[1] K. Yoshida, Y. Sato, K, Shoji and S. Abiko. Dynamics Simulation of a Manipulator System Mounted on a Space Station and Improvement of the Operational Performance by Using Redundancy. *In Proc. of ROBOMEC 2005*, pages 227-232, July 2005.

[2] NASA/JPL. Mars Explorations Rovers. *Available: http://marsrovers.jpl.nasa.gov/* (2006, March).

[3] S. Abiko, K. Yoshida, K. Shoji and Y. Sato. Performance Improvement of JEMRMS in Light of Vibration Dynamics. *In Proc. of International Symposium on Artificial Intelligence, Robotics and Automation in Space*, pages 1-8, September 2005.

[4] The World Wide Web Consortium, Extensible Markup Language, Available: http://www.w3c.org/XML (2006, March).

[5] M. Makatchev and S. K. Tso. Human-Robot Interface Using Agents Communicating in a XML-Based Markup Language. *In Proc. of 2000 IEEE International Workshop on Robot and Human Interactive Communication*, pages 270-275, September 2000

[6] M. Fujita and K. Kageyama. An Open Architecture for Robot Entertainment, *In Proc. of Int. Conf. on Autonomous Agents*, pages 435-442, 1997.

[7] T. Yoshimitsu, et al. Autonomous Navigation and Observation on Asteroid Surface by Hopping Rover MINERVA. *In Proc. 6th Int. Symp. on Artificial Intelligence, Robotics and Automation in Space*, June 2001.

[8] K. Yoshida, T. Kubota, S. Sawai, A. Fujiwara, M. Uo. MUSES-C Touch-down Simulation on the Ground. *In AAS/AIAA Space Flight Mechanics Meeting, Paper AAS 01-135*, February 2001.

[9] M. Chacin, and K. Yoshida. Multi-Limbed Rover for Asteroid Surface Exploration Using Static Locomotion. *In Proc. of International Symposium on Artificial Intelligence, Robotics and Automation in Space*, pages. 1-8, 2005.

[10] M. Chacin and K. Yoshida. Evolving Legged Rovers for Minor Body Exploration Missions. *Proceedings of the 1st IEEE / RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics*, February, 2006.

[11] M. Chacin, K. Nagatani and K. Yoshida. Next-Genenation Rover Development for Asteroid Surface Exploration: System Description. *In Proc. of the 25th International Symposium on Space Technology and Science and 19th International Symposium on Space Flight Dynamics*, (to be published 2006).

[12] M. Chacin, J. Merino, K. Yoshida and H. Yano. Bio-mimetic Mechanism for Rovers Under Microgravity Surface Environment. *In Proc. of 14th Workshop on Astrodynamics and Flight Mechanics,* pages 227-232, 2004.

[13] E. Rohmer, T. Takahashi, and E. Nakano. Synchronization of Modules in a Distributed Telerobotic System for Construction Machines. *In Proc. of the International Engineering Research Conference 2005*, pages 105-11, March 2005.