ADVANCED SPACE ROBOTICS SIMULATION FOR TRAINING AND OPERATIONS

Xavier Cyril, Gilbert Jaar, Jean St-Pierre
Space Systems, CAE
8585 Côte de Liesse, P.O. Box 1800
St. Laurent, Québec H4L 4X4
Tel: (514) 341-2000
Email: cyril@cae.ca, jaar@cae.ca, jstp@cae.ca

This paper describes the advances being made in space robotics simulation to meet the challenges of astronaut training and space operations support. This simulator (MOTS) is being used to train International Space Station astronauts to perform on-orbit robotics tasks. It also supports mission planning and task verification in an operationally representative environment. The simulator supports critical tasks to be performed by astronauts including payload handling, berthing and de-berthing. MOTS is a state-of-the-art simulator providing astronauts with a simulation representative of the space station dynamics and visual environment. It provides real-time high-fidelity simulation of the flexible dynamics performance of two robotic arms (space station arm and shuttle arm) concurrently in a micro-gravity environment to support complex "hand-off" tasks. Contact dynamics models have been added to enhance the realism of berthing payloads to the Space Station with multiple contact points simultaneously tracked. 3D visual models support realistic views generated by the space station cameras in an operational and dynamic lighting environment that includes the production of split screen views. The incorporation of the Space Station Robot Arm Flight Control System Software provides an invaluable and confident environment in which on-orbit tasks is being planned and practiced. MOTS is also being integrated into several facilities at the Canadian Space Agency such as the Space Operations and Support Centre, to support on-line diagnostics.

Introduction

Space robotics provides a cheaper alternative to perform space systems assembly and maintenance, which otherwise would have to be performed by Astronauts during a space walk. Based on the success of the space shuttle robotic arm, the next generation robotic arm is being developed for use on the International Space Station. This arm which is being supplied by the Canadian Space Agency (CSA) will be used to perform tasks, such as, capture payloads (space station components), maneuver them out of the space shuttle payload bay and berth them to the partially complete space station. These tasks and others require dexterity in the robotic arm and considerable skill from the astronaut. The astronauts will not have direct view of the arm and its environment. They will be operating the arm from within the space station module via 3 views provided by the cameras mounted on the robotic arm and the space station. Though these cameras are an improved version of the space shuttle camera they still exhibit artifacts commonly seen in all CCD cameras.

Additionally, the 55 feet long space station robotic arm will oscillate under the combined inertia of the arm and the payload. Operating the arm under these conditions is a challenging task. In order to prepare the astronauts for this task a high fidelity simulator is needed, which will faithfully replicate the dynamics of the arm and the environment around it. Such a simulator has been developed for the CSA to support space robotic operations planning and training.

In this paper we describe the simulator, which is also referred to as MOTS. The simulator is accurately described by users as a world class simulator providing astronauts with a simulation representative of the Space Station environment through re-creation of the on-orbit dynamics and visual environment. The simulator supports the real-time high-fidelity simulation of the flexible body dynamics of two robotic arms concurrently operating in a micro-gravity environment in sufficient detail to support complex tasks. Contact dynamics models have been added to the simulator architecture to support the berthing of payloads to the Space Station with multiple contact points simultaneously tracked. Realistic views that include split screen presentation and dynamic overlays are

generated from 3D graphic models using MSS camera models to represent the operational environment. The incorporation of the Space Station Robot Arm Flight Control System Software to enhance the confidence in the simulator representation of the "as-built" flight system is currently nearing completion. Another important and challenging technical advance is the interface of the simulator with a real robot in which the simulation drives the robot through a hardware controller and contact forces are fed back to the simulation to close the loop and accurately represent contact dynamics.

Overview of MOTS

The simulator provides users with a representative environment of the Robotics WorkStation (RWS). In addition, ground support staff extensively utilize the simulator to prepare and conduct tasks under nominal and malfunction operating conditions. High fidelity software models enable the representation of flight system behavior. The simulation models provide system responses through a flight-representative human computer interface. A realistic simulation of the operating environment is also provided through the reproduction of appropriate lighting, audio, and video performance. Video images are rendered from a database of the Space Station visual objects with a customized high performance visual renderer.

The simulator supports operations analysis by providing a sophisticated set of real-time simulation and off-line data collection, conversion, and analysis tools. These tools are used to verify system operating procedures, on-board displays, telemetry definitions, system models, and performance data. The data recording and analysis function enables the user to analyze anomalous behaviour and isolate likely causes of equipment failure. The simulator also generates accurate command and telemetry data streams. These data streams are also used to support validation of the ground segment control, monitoring, and diagnostic facilities.

The main hardware components include a simulation host computer, an RWS, an instructor station, and a video distribution and recording system.

Following an extensive trade-study and benchmark investigations, the computer platform selected as the primary simulation host was a Silicon Graphics Onyx™ equipped with 12 R-4400 CPUs and 2 graphic pipes to run the visuals. This prime host system was recently upgraded to 20 R-4400 CPUs and 3 fully populated Reality Engine™ graphics pipes to enable the support of additional manipulators and camera views.

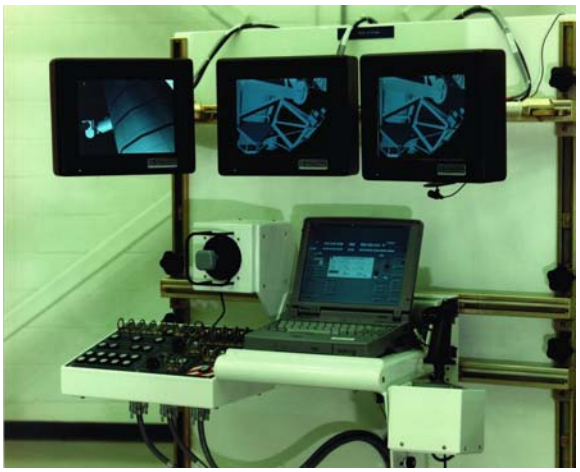

Figure 1: MOTS Robotics Work Station

The RWS, shown in Figure 1, provides crewmembers with the capability to perform all MSS commanding. This includes video system commanding and the ability to route MSS, Station, and Orbiter camera views to the three video monitors. The users interface with the simulator via on-orbit Graphical User Interface pages displayed on a laptop computer. They control the simulation via a set of 3-degree-of-freedom hand controllers (one translational and one rotational) and a display and control panel in addition to the laptop.

Figure 2: MOTS Instructor Station

The instructor station, shown in Figure 2 provides the capability for instructors to select, load, monitor, and control all aspects of training sessions including the ability to insert and manage malfunctions. A set of hand controllers is available to control the robotic manipulators. This station provides the instructor with an arbitrary viewpoint of the simulated scene that is positioned from a control page or by using the handcontrollers.

The simulator is built around CAE's off-the-shelf simulation environment, which provides the capability to define, build, manage and control real-time simulations and is used in flight simulators all over the world. The real time dispatcher schedules the execution of the models and processes in a specified order at a specified rate. This dispatcher handles both synchronous and asynchronous processes.

The software development environment features a suite of integrated tools to support the development of new simulation scenarios. It is integrated within the CAE simulation environment and is complimentary to the main simulation environment.

The record and playback function captures key simulator data, which enables the scene to be replayed within the simulator. It can record up to 4 channels of video and audio simultaneously on tapes, which can be replayed later either in real-time, frame-by-frame, or slow motion. The simulation can be restored to any point in the sequence, as controlled from the playback control page.

The malfunction insertion capability during a simulation session is another important function of the simulator. Specified malfunctions can be activated or deactivated manually, or based on time or event. The malfunctions are stored in a malfunction database, which contains all the malfunctions and triggering parameters.

Simulation Models

The simulation models are hosted on the computer platform and communicate with each other via the shared memory segment. In order to achieve an integration update rate of 1000 Hz, the models are parallelized across 4 CPUs. The simulation model processes are synchronized to ensure that the system is deterministic and runs in real time.

Simulation models include nominal and malfunction behavior through faithful representation of equipment performance and are divided into two major categories: physical models and equipment models.

The physical behavior models consist of a flexible multi-body dynamics model used for payload captures, berthing and handoff. Currently three robotics arms are simulated; namely the Space Station Remote Manipulator System (SSRMS), the Shuttle Remote Manipulator System and the Special Purpose Dexterous Manipulator (SPDM). The simulator also provides a power distribution model that determines the power status of the equipment, and a thermal model that computes the temperature of each MSS component based on the heat generated and dispersed.

The equipment models consist of MSS equipment hardware control models and on-board software models. One example of such a model is the simulation of the software that resides in the robotics workstation. The model provides a 2-way communication between the on-orbit control pages, control panels, and handcontrollers, and the simulated MSS elements. Commands are forwarded to the lower level equipment models and telemetry and status information is returned to the robotics workstation.

A very important addition to the simulator is the incorporation of contact dynamics that provides a realistic payload capture and berthing environment. A separate contact dynamics model supports each of the two possible types of collision scenarios. The Continuous Impulse Balance Method (or "impulse-based model"), described in [3] for a two-body system, has been generalized to support robotic systems composed of multiple flexible bodies to support general contact. The Hertz compliance method (or "Hertz-based model"), based on a toolkit provided by MD Robotics as described in [4], has been refined to accommodate the processing requirements of a real time environment and supports multiple point contact within defined berthing regions only.

Collision detection is fundamental in the overall solution of a contact dynamics problem and interacts closely with the geometric modeling of simulation objects. Two models are implemented within MOTS. The impulse-based model shares with the graphics animator the geometric database that is used to draw objects. This database is generated by a commercial graphics utility. This feature makes it possible to maintain a single repository of geometric information for all simulation bodies under configuration management. Furthermore, since what you see is what can collide, and collisions between any two (or more) objects are readily accommodated. An algorithm derived using techniques from computational geometry determines whether two objects are colliding. It returns the location of contact on the colliding bodies and the local normal to the dynamics simulation. This information is used by the impulse-based model to calculate the contact forces based on the normal approach velocity between bodies and on the estimated coefficient of restitution and duration of impact. The analysis factors in the external and control forces to allow sustained sliding contact where appropriate.

The impulse-based model is the method of choice for off-nominal, undesirable collision situations; i.e., not planned berthing sequence scenarios. The user is notified when off-nominal contact occurs and the dynamics simulation engine realistically renders the robotic system response. One drawback to this method is the limitation imposed by the accuracy with which the local normal at the point of contact can be calculated. Indeed, this is done numerically for example, by averaging the normal vectors of the polygons involved at the location of contact. When sharp edges are present the resulting normal may be such that sliding contact, which takes place in the plane perpendicular to the normal, has one body penetrating into another.

The Hertz compliance method implemented in MOTS, on the other hand, is supported by a separate geometric database of simulation objects. Contact surfaces are modeled as linear or quadratic, and collision detection correspondingly requires solving the linear or quadratic programming problem. For this approach it is necessary only to model the berthing interfaces of payloads. This method determines the interference between bodies, and uses Hertz contact law to calculate the forces. The performance of the Hertz-based model in the presence of multiple contacts and sliding interfaces is very good, and it is used whenever added fidelity in the modeling of complex berthing scenarios is desired.

Flight Software in the Loop

MOTS is required to simulate several control systems present in the MSS, at the different subsystem levels:

- Joint Control Software (JCS), the low-level joint controller
- Latching end-effector Control Software (LCS), the low-level latching end-effector controller
- SSRMS Arm Control Software (SACS), the mid-level arm controller
- Operations Control Software (OCS), the high-level MSS controller

When MOTS was initially started, the MSS flight software components were still under development. Some "engineering models" of these components, which were used for design validation, were selected for initial MOTS capability. These engineering models were at a sufficient level of fidelity to provide initial crew training.

In its long-term planning, the Canadian Space Agency had determined that MOTS should contain as much actual flight software as technically and economically feasible. The integration of actual flight software in a simulator has several advantages:

- increased level of confidence in simulator fidelity
- ease of simulator upgrade for new releases of flight software
- simulator can be used for testing flight software in complex scenarios

However, as the reader may already know, there is a cost and risk factor associated with trying to integrate any kind of real-time flight software in a simulation environment.

The first difficulty is in the reusability and modularity of the flight software code. It is generally good programming practice to develop flight software by separating it into several abstraction layers, such as a hardware interface layer, services layer, executive layer and application layer. The number of flight software layers integrated in the simulator usually reflects on the desired level of fidelity required and on technical feasibility of each option. To integrate all layers, for example, it would be necessary to provide a complete software emulation of the original processor instruction set. This is quite onerous and not necessary for an operations and training simulator (though a verification and test facility may need it).

For MOTS, it was determined that the most cost effective approach was to encapsulate the highest-level application layer into the simulator executive and host computer services. For the encapsulation of the SACS flight software, a wrapper module was provided by MD Robotics, and an interface module was developed by CAE to interface the control system with the rest of the MOTS models. This interface module is responsible for mapping data between the simulator models shared memory variables and the Ada records passed as arguments to the SACS wrapper.

The second difficulty resides in the programming language itself. The portability of the language in which the flight software is written, and the availability of a suitable compiler on the simulation platform are important factors to consider. In our case, the flight software was written in Ada. The Ada language is one of the most portable ones, since the syntax is well defined (though there are two standards, Ada83 and Ada95) and there are no assumptions made with regards to data types and data representation, which are the usual stumbling blocks for cross-platform porting.

After evaluating several compilers available for our simulation platform, it was decided to select the SGI Ada95 V1.3 compiler, which is based on GNAT 3.11b.

The main reason for choosing that solution was that the GNAT environment permitted us to easily integrate the compiler inside CAE's simulation build tool (called SIMex). This was not the case with other Ada compilers, which had proprietary mechanisms for building and maintaining executables, and which were incompatible with the SIMex configuration management functions.

Though the compilation of the SACS flight software was mostly uneventful, we did incur three run-time problems that made the simulation crash. The first problem was with the usage of "records of records". We had to modify the flight software to write into the record element-by-element instead of copying the record as a whole. The second problem was with the usage of the "use at" clause which creates a pointer to an address. Here we had to modify the flight software to introduce an intermediate variable that contained the address that was pointed to directly in the original code. The third problem was a still unexplained crash when calling one of the functions provided by the flight software itself. In this case, we had to inline the offending code in the calling program as a temporary workaround. It is expected that future compiler releases will eventually fix these problems.

Advanced Visualization

Within the Space Station the astronauts will operate the robot arm from one of two Robotics WorkStations, where camera views are displayed on three 11.4" flat panel LCD monitors. Without direct vision the astronauts will be challenged to operate the arm mainly through feedback provided by these camera views. These will tend to introduce artifacts, which make the task more demanding. The simulator generates views that are representative of the images produced by the flight cameras to support astronaut familiarization and training, or preparation for on-orbit tasks.

To generate these images in real time with the computer technology available was perhaps the greatest challenge we faced. To add realism to these images required the incorporation of shadowing and camera effects such as blooming, smearing, depth-of field, and out-of-focus. To meet these requirements CAE developed a real time visual renderer that supports all of these camera effects, the end result of that is a simulator that provides the astronauts with realistic camera views rendered at a nominal rate of 15 to 20 Hz. The visual renderer relies upon the visual database and the data provided by the software simulation models. The visual database consists of graphical models of the Space Station and MSS elements (Figure 3), constructed for efficiency and organized into several levels of detail. The database when rendered provides an impressive lighting environment that faithfully emulates that in space. The latest addition to the visual model is the capability to multiplex 2 views on the same monitor. This enables the user to have as many as 5 views on the 3 monitors of the Robotics workstation.

Figure 3: International Space Station

In order to run the simulation models on one platform and the visual model on another platform, a High Level Architecture protocol was implemented in the simulator. The protocol is a general-purpose architecture for simulation re-use and interoperability and is in the process of becoming an open standard through the Institute of Electrical and Electronic Engineers. This will significantly improve the visual update rate and speed up the system load time.

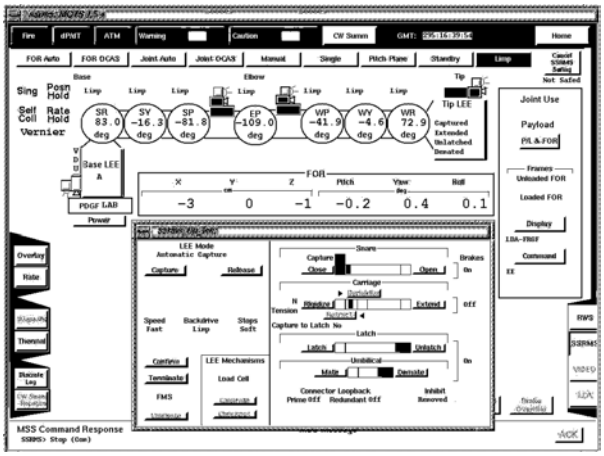

Figure 4: On-orbit HCI page

The primary method by which users interact with the MSS equipment on-board the Space Station is the

human computer interface pages (Figure 4) hosted on the Personal Computer System (PCS) laptop. The implementation and activation of these pages within the simulator was another significant challenge. To make these pages "come alive", the CAE design utilized Interface Control Documents to determine the correct scenario and derive the internal logic. Interfaces from these pages to the simulated models were determined from control system design documents. Currently the Space Station and the Orbiter arm control pages are created using an off-the-shelf package known as SAMMI, however the SPDM control pages are being implemented in C^{++}. These pages have yet to include a real world interface based on a 1553 protocol.

Hardware in the Loop Simulation

The simulator is used to support the emerging SPDM Task Verification Facility (STVF) which embodies hardware-in-the-loop simulation. Within this facility a real robot will reproduce the Special Purpose Dexterous Manipulator end-point motion resulting from real contact forces between mock-ups of the SPDM payloads, known as Orbital Replacement Units (ORUs), and the work-site. The real contact forces will then be fed back to the simulator, which will be running the SPDM dynamics and control system models. Data will be exchanged between the simulator and the robot on a firewire, since the data exchange must be at a rate of 1000 Hz. To achieve real-time performance with the added simulation models within the available resources, a parallel simulation architecture was implemented.

The simulator will also be interfaced with the Space Operations Support Centre to drive it with simulated telemetry data. This will enable responses to simulated MSS behavioral data to be validated and compared with actual telemetry and command responses. The simulator will also continue interfacing with the Artificial Vision Unit (AVU) that locates objects by tracking appropriately positioned surface mounted targets. The AVU supports many Space Station tasks where fine positioning takes place out of the line of sight of the operator and available cameras. The simulator is also capable of supporting concurrent simulations where flight activities are shadowed in near-real time and corrected with flight data telemetry as it is received. With this capability ground support personnel will possess a powerful tool to analyze and support flight robotic tasks as they unfold.

Summary

The simulator plays a key role in preparing astronauts for the operation of the MSS and verifying operations procedures. It provides a faithful simulation of MSS equipment and its environment through high fidelity simulation models and visual simulation. The simulator is hosted within the CAE real-time simulation environment, which has an excellent track record and is proven to provide deterministic and consistent simulations in flight simulations worldwide. The simulator was used by astronauts and has gained their acceptance as a world class simulator. However, there remain features to be added in the near future which require full utilization of current technology. Additionally, the simulator will be validated against available "as built data" and ultimately against flight data. The simulator is established upon a solid technical and architectural foundation and has proven capable of supporting a variety of user needs. This capability will be enhanced with the incorporation and validation of many new capabilities in the near future.

References

1. G. Jaar, X. Cyril, D. Harvie and D. McCabe, "MSS Operations and Training Simulator", *Astro 98 Conference*, Ottawa, Ontario, 26-28 Oct 98.
2. G. Jaar, "End-to-End Simulation of the MSS within MOTS", *Symposium on Space Mission Operations*, St-Hubert, Québec, 4-5 Dec 1995.
3. F. Belanger and X. Cyril (1997) "Continuous Linear Impulse Balance Method for Real Time Simulations of Contact Dynamics", *Proceedings of the AIAA Modeling and Simulation Technologies Conference*, AIAA Paper 97-3670.
4. M. Ou (1995), "Contact Dynamics Modeling for the Simulation of the Space Station Manipulators Handling Payloads", *IEEE International Conference on Robotics and Automation*, pp. 1252-1258.