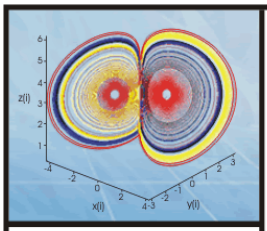

7th Brazilian Conference on Dynamics, Control and Applications

May 07 - 09, 2008
FCT - Unesp at Presidente Prudente, SP, Brazil

SPACE TELEOPERATION: A SOFTWARE SIMULATION AND CONTROL APPROACH

Jonas Gentina [1], *Ijar Milagre da Fonseca* [1]

[1] Instituto Nacional de Pesquisas Espaciais, Divisão de Mecânica Espacial e Controle, São José dos Campos-SP, Brasil, jonas@dem.inpe.br, and ijar@dem.inpe.br

Abstract: This paper presents an overview of the technological approaches used in the space teleoperation scenario, emphasizing the computational and software capabilities that enable real time simulation and control of complex space activities. The work also includes the technical issues involving supervised autonomy for space telerobotics and for collision prevention.

Keywords: space teleoperation, telerobotics, real time simulation.

1. INTRODUCTION

The nowadays space operations to implement spacecraft tracking and control are mostly performed remotely. Such teleoperations include data transmissions of events, ranging from simply vehicle component telemetry data to complex mission system information links, such as image downloads from high-resolution cameras or even onboard software updates.

All of those operations are always accomplished under management of very robust and reliable software systems, implemented both at ground stations and at spacecraft via onboard-embedded software. This paper discusses the most used techniques implemented to perform teleoperations in the space environment, especially those on autonomous control of unmanned space vehicles and space telerobots. The work also explains some of telerobotics procedures to provide on-ground and on-board autonomy supervision and technical application for some of the most critical and risky space maneuvers. These maneuvers require several steps, for which techniques for collision detection and avoidance, impact stability and force control of the space manipulators must be tested and verified. Some of the technological approaches related to telerobotics procedures are presented here, mainly those focusing on software systems that enable those space operations at several different phases.

The current literature presents descriptions of some useful systems that can be built around the concept of humans-supervising telerobots. Based on this approach it is possible to develop space telerobots that perform low-level operations automatically. Low-level operations refer to the basic tasks that can be accomplished by the robot without any human operator action. Behind the scenes there are many methods for space telerobotics supervised autonomy. Supervised autonomy is also a viable near-term approach for the remote control of space manipulators. In this context the operation safeness is achieved by generating commands with specific parameters to each task and assessed by a priori simulation. Regarding this approach, actually there exists the possibility to simulate dynamical and complex real time systems into virtual environments. To improve the system performance, the simulation can be run in a distributed manner by the use of a modern architecture capable to support such distributed or parallel software system.

In this paper the space teleoperation and the space telerobotics techniques as well as their applications with the software systems concerning real time distributed simulation are compared. The section 2 presents an overview of space telerobotics and space teleoperated activities. The section 3 presents the concepts related to the supervised autonomy for space telerobotics. Section 4 approaches the techniques for collision prevention, stability and force control by space manipulators. Then, ideas related to the pre-needed efforts to recreate real-time simulation scenarios using parallel and distributed systems and architectures are shown in section 5.

2. OVERVIEW OF SPACE TELEOPERATION

Since 1957, when the launching of Sputnik pushed the world into the space age vast resources have been invested in developing space systems. Those investments have been enormously successful. Earth-orbiting satellites have revolutionized communications, intelligence gathering, weather prediction, resource management and navigation. Scientific satellites have provided a wealth of data that has significantly improved the scientific understanding of the Earth, the solar system, and the universe [1].

Successfully manned missions, as Apollo that has taken the man to the Moon, and the Space Shuttle, whose crew has refurbished the Hubble Space Telescope in December 1993, have demonstrated that astronauts can perform assembly, maintenance, and repair operations in space. However, the use of astronauts on a large scale for such operations is far too costly and entails significant safety risk. Teleoperated unmanned space vehicles, also known as space telerobots, can extend astronaut capabilities and performance, thereby increasing mission performance and reducing costs.

Telerobots can be roughly described as "machines that perform physical tasks" [1]. The motivation for using them in space is to accomplish tasks as inspection, maintenance, repairing, module changing/replacing, cleaning up, assisting science and technological experiments, performing repetitive operations, and capturing and despinning satellites, among other.

Ruoff [1] explains that an exclusive feature makes the difference when classifying telerobots. That feature is associated with the remote telerobot capability of their control system, in the sense that the human operators and ground support along with the remote telerobots are capable of accommodating uncertainties. In other words, the system must be able to determine the state of the task and relevant objects, iteratively determine the actions to take, predict their effects, and coordinate the subsystems to perform the actions while monitoring their effects to make sure they are consistent with the predictions. If the predicted and observed effects are not consistent, there is a potential problem. Finally, the control system must be able to monitor and maintain the own system health.

Satellites and spacecraft have been restricted to operate in a free space in the sense that obstacles do not play an important rule in the scenario of space operations and maneuvers. In such an environment the control objectives can be in some aspects easily characterized. Such systems do not have or do not need the ability to sense and classify complex external situations and make quick onboard decisions. They operate in an open-loop manner for long periods and most of the control decisions are made on ground. On the other hand, rovers and space telerobots must be capable of performing mechanical operations at reasonable rates in complex natural environments.

Despite the massive advances in improving the machine intelligences, human judgment is still essential for difficult situations. In this case the lower level behavior of telerobots such as the moderately complex sensor data interpretation and the motion/force control are easily automated. According to Ruoff [1], a practical approach to develop useful telerobot systems can be stated as: 1) automate lower level functions by developing reliable control algorithms that adapt on the basis of sensory information, 2) rely on human operators for providing overall task guidance and supervision, and for handling special situations, and 3) develop advanced interfaces and tools that aid in planning and managing telerobot tasks and permit the operator to communicate easily with the system at multiple levels of detail.

2.1. Space telerobotics systems

A space telerobot comprises one or more manipulators, each with several degrees of freedom, mounted on a platform which might be mobile (or free flying). A space telerobot also has a sensor suite, usually including arm, platform, and mobility state sensors, force sensors, cameras, and necessary computation and support systems. The telerobot's major subsystems are computing, coordination, external sensors, manipulation, mobility, payload, perception, platform, power, telecommunication, thermal control, and the telerobot executive. These subsystems are described further in Fig. 1.

In the present work, it will be taken into consideration the computing, coordination, perception and executive subsystems, since they are all in some means interconnected together. The most important subsystem in the approach of this paper is the computing one. It is under the control of the telerobot executive and aggregates the computational devices aboard a telerobot, including general and special purpose computers, low level controllers, sensor preprocessors, and other dedicated electronics. All control resides in the computing subsystem. The telerobot executive, perception, and coordination subsystems reside in the computing subsystem as well.

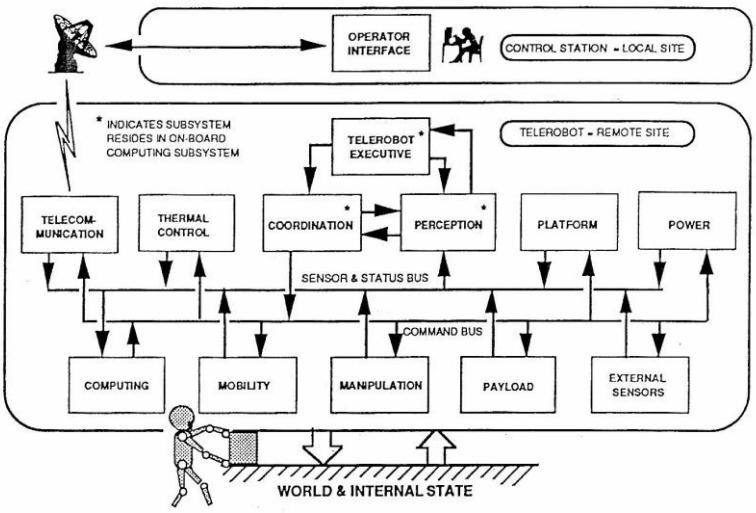

Fig. 1. Space telerobot system showing command and data flow (C. F. Ruoff [1], p. 6)

The coordination subsystem, which performs kinematic and dynamic computations as necessary and coordinates the behavior of the various subsystems and devices that are under the control of the executive, resides in the computing subsystem as hardware and/or software. It receives state information from telerobot actuators and from external sensors (interpreted by the perception subsystem) and can send, in advanced systems, predicted state information to the perception system so as task evolution consistency can be determined, that is, the system can determine if the task is proceeding as predicted.

The perception subsystem receives input from the various state and external sensors as well as from the telecommunication system. In sophisticated telerobots it computes a summary of the external environment, telerobot, and task states, which is used by the telerobot executive and the coordination subsystems. In simple systems, the perception subsystem may just perform transformation on sensory data.

The telerobot executive, which also resides in the computing system schedules and controls the overall high-level behavior of a telerobot's subsystems, except for automatic fault protection and reflexes. It receives operator commands and instructions as well as world, telerobot, and task state information from the perception subsystem. In advanced telerobots the executive includes planning, reasoning, behavior prediction, and fault diagnosis tools. The executive can issue commands to both the perception and coordination subsystems.

A typical space telerobot system also includes a control station and the telerobot itself, shown schematically in Fig. 1, along with command and data flow. The control station is also called as "local site", and includes the interface which the operator uses for both to comprehend the remote task and to control the telerobot. The control station communicates with the telerobot through a data link. The ground may include support of powerful simulation and planning computer capabilities. The operator supervises the telerobot tasks, resolving difficult situations and determining

what routines or macros (series of commands) to use. This operator can specify a task and then relinquish control to the telerobot, which returns control when either the task is complete or an impasse is reached. The operator can also seize the control at any time. Thus, control is traded back and forth between the operator and the telerobot.

The telerobot is also called as "remote site", and physically performs tasks under the control of an operator. Teleoperators can interact with the telerobot systems in the form of master-slave systems, where the master is a local-site replica of the remote-site slave. In this scenario the operator performs a task moving the master by watching a visual representation of the remote worksite, as if the master were performing the task itself. According to that approach, it is necessary the teleoperators to provide telerobots with detailed commands, either in the form of macros or in the form of motion commands. Human operators must be prepared to assist them in locating and identifying objects. In this line, Hartley and Pulliam [2] have developed an experiment in which heads-up displays and voice input/output were implemented on an experimental pilot console. Their main objectives were to supply optimized future operations of remotely pilot vehicles and the ISS operations.

In non-replica master-slave systems the master and slave are not geometrically similar. In all-software teleoperated approaches, the master use to be a computational-mathematical model representation of the slave. In such systems, axis coordination is handled by a computer in the control-loop that continually maps the present position of the master handle into the (scaled) position of the slave hand. The computer uses master kinematics to calculate the Cartesian position of the master handle in space and slave kinematics to transform this position into position commands for the slave axes, thus making the slave hand perform the same (scaled) Cartesian motion as the master handle. Error and contact signals are used to back drive the master, giving a sense of contact with the remote environment. Those procedures are illustrated by Fig. 2.

Controlling space telerobots from the ground could make them extremely attractive, but ground control places stringent demands on their control systems because of communication delays, data rate limitations, and task uncertainties. To compensate those constraints while preserving performance, telerobot systems will need greater intelligence and autonomy, which are independent concepts related to environmental and task complexity that have a profound impact on telerobot system performance. Extremely intelligent robots are far beyond the state of the art, but useful systems can be built around the concept of humans supervising telerobots. Such an approach permits robots to perform low-level operations automatically while freeing human operators to concentrate on higher level task elements. As autonomous system technology advances, it will be possible to delegate higher levels of decision making to telerobots, reducing the load on human operators, ground control, and telecommunication systems while improving telerobot performance. In this way future space telerobotics will certainly be able to develop advanced machines useful for excavation, construction, assembly, maintenance, inspection, calibration, repair, solar array emplacement, and habitation construction, and site preparation of manned outposts or observatories at the Moon and Mars, for instance. Space telerobots are also planned to do dangerous, risky or costly space activities, as those that handle with nuclear reactors, cryogens, and toxic propellant onboard spacecraft.

3. SUPERVISED AUTONOMY FOR SPACE TELEROBOTICS

Space applications provide both an important purpose for telerobotics and many important constraints on its implementation approach. Currently, the spacecraft designs are developed to be reliable and fail-safe, so their systems are optimized to provide only the required most critical needs to achieve mission success. Those needs include mostly the onboard capabilities for communication and control of the spacecraft and instruments. The Earth-based segment of the teleoperated systems generates command sequences which are telemetered to the remote spacecraft.

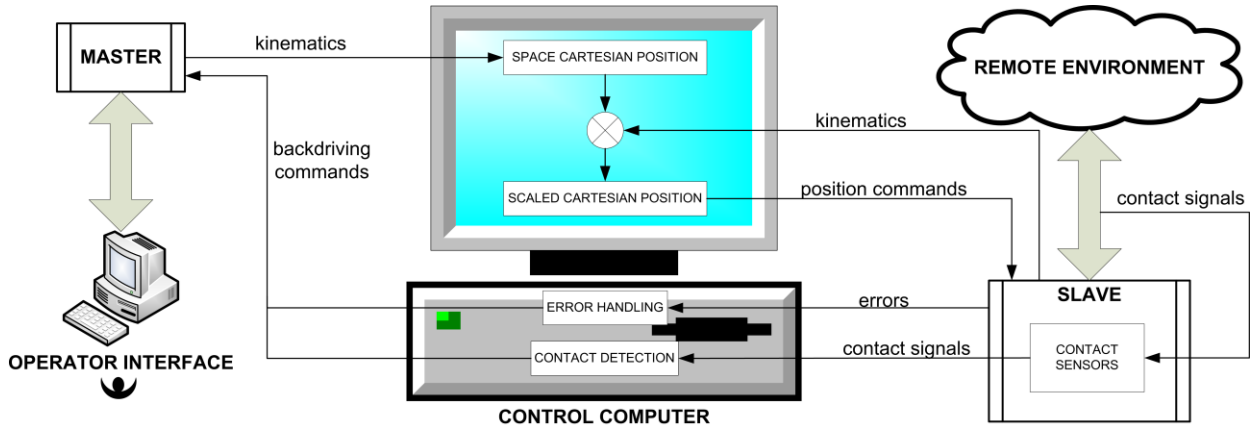

Fig. 2. Teleoperated master-slave systems schema

Command generation on the ground, based on updated data from the spacecraft, provides the needed system flexibility to achieve mission success. Extensive human and computational resources are much more complex on ground stations than those aboard the spacecraft. As seen in the previous section, the spacecraft is able to execute command sequences, which have been telemetered from Earth as well as to react to anomalous situations. Ground-based control of remote unmanned spacecraft is an application of supervised autonomous control.

Telerobotics methods can be separated into three types [3], as seen by Fig. 3: manual control, supervisory control, and fully automatic control. In manual control, all the robot motion is specified by continuous input from a human, with no additional motion caused by a computer. In supervisory control, the robot motion may be caused either by human inputs or computer-generated inputs. In fully automatic control, all the robot motion is caused by computer-generated inputs.

There are two primary subsets of supervisory control: shared control and supervised autonomy. In shared control, the operator commands are sent during the execution of a motion and are merged with the closed-loop motion generated automatically. In supervised autonomy, the autonomous commands are generated through human interaction. However, the commands are sent for autonomous execution remotely. A command can be sent immediately or iteratively saved, simulated, and modified before it is sent for execution on the real robot systems.

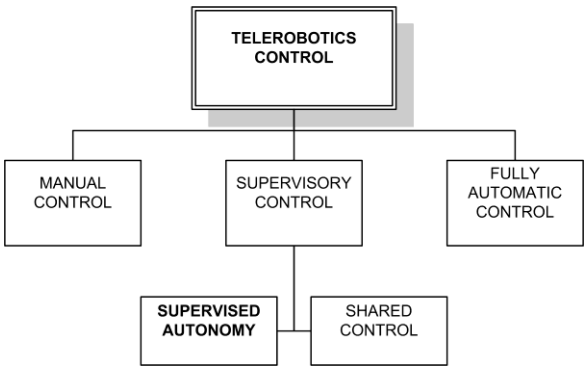

Fig. 3. Methods of telerobotics control

The ability to iteratively save, simulate, and modify commands before sending them for execution is a critical feature of supervised autonomy which distinguishes it from other forms of supervisory control. For safety purposes it is important to be able to simulate task execution before sending command sequences to the manipulator for task execution. Safety is achieved by verifying the commands before sending them for execution on the real robot systems and through real time monitoring. Real time simulation and monitoring are discussed further in the last section. Commands can be modified and simulated until they are acceptable for execution by the telerobot. Individual commands can be concatenated into a command sequence (macro) which can then be iteratively simulated, correctly modified and inserted into a large sequence. Command Sequence generation for autonomous spacecraft is a formal process because dangerous or incorrect commands could result in serious damage, loss of unique scientific opportunities (e.g., during a planetary flyby), or loss of the entire spacecraft, ruining the mission. In shared control, the operator commands are sent immediately to be merged with autonomous execution. Safety in shared control is achieved either by relying on the operator to input safe motions, or by having real time autonomous monitoring and modification of the motion specified by the operator.

One of the most famous situations where a wrong command was issued to a spacecraft causing a mission failure happened on the Mars Climate Orbiter in 1999 [4]. The spacecraft was one of the Mars Surveyor ’98 program and was intended to enter in orbit of the red planet at an altitude of 140–150 km. However, a navigation error caused the spacecraft to reach as low as 57 km above the surface. The spacecraft was destroyed by atmospheric stresses and friction at this low altitude. The navigation error arose because a NASA subcontractor used imperial units (pound-seconds) instead of the metric units (newton-seconds), as specified by NASA. The problem was due to the spacecraft flight software errors, partly because the software had been adapted from use on the earlier Mars Climate Orbiter, without proper testing before launch, and partly because the navigation data provided by this software was not cross-checked while in flight.

Another well-known mission failure occurred in June 4, 1996, when the flight 501 of the European Ariane 5 rocket veered off its flight path 37 seconds after launch. The launcher was destroyed by its automated destruct system when high aerodynamic forces caused the core of the vehicle to disintegrate [4-5]. The reasons of the failure were quite the same as those reported by the Mars Climate Orbiter, that is, a malfunction in the flight control software. The Ariane 5 software reused the specifications from the Ariane 4, but the Ariane 5's flight path was considerably different and beyond the range for which the reused code had been designed. Specifically, the Ariane 5's greater acceleration caused the back-up and primary inertial guidance computers to crash, after which the launcher's nozzles were directed by spurious data. Pre-flight tests had never been performed on the re-alignment code under simulated Ariane 5 flight conditions, so the error was not discovered before the launch. The error is featured as one of the most infamous computer bugs in history.

Disasters like those could be easily avoided just by the use of a prescription described by the supervisory control methods, more specifically, the supervised autonomy. As mentioned in the first section, flight systems require robust flight qualified software running in limited computing environments (limited compared to the ground systems). Modification of flight software during flight, although possible, requires an extensive and costly qualification process. Therefore, to prevent undesired situations such as those just described, the solution of supervised autonomy taken for unmanned robotic spacecraft control are described here.

Another important feature of supervised autonomy is the bounded behavior execution. Bounded behavior execution allows task execution to diverge from the nominally planned motion within a specified bound. Since the remote environment cannot be known exactly a priori, real time execution will rely on both the pre-planned trajectory and perturbations computed via remote sensed data. Thus, the safety of execution within a specified bound can be tested a priori at the local site. The remote system can then

autonomously monitor execution in real time to ensure that the state of the motion is within the specified bound. If execution moves out of specified bounds, then an automatic reflex action is invoked and further local-site commands are awaited.

3.1. Supervised autonomy systems

The local and remote components of a supervised autonomy system can be divided into subcomponents. The local site includes sequence generation, sequence analysis, monitoring, and telemetry. The remote site includes telemetry, command parsing, sequence control, real time control, monitoring, and reflex. The command sequences are composed of command types and associated data which specify the desired spacecraft and instrument control behavior. The flight software is fixed but provides general command types which can be parameterized to generate a wide range of specific control behaviors.

Sequence generation is the process of generating a command sequence which can be telemetered to a remote autonomous robot control system. An operator interface is provided and the operator uses this interface to specify the desired commands. Computer aids can also give support in the specification of tasks, commands, and parameterization. Computer aids include modeling, visualization, and task planning. Computer modeling provides a model of the manipulated systems or the task execution environment. The model can even be modified to match the remote scene using data returned from the remote environment. Visualization provides a graphical representation of the scene. An accurate representation of the task execution scene is important to ensure that the priori simulation is a valid representation of the execution that will occur on the real robot. Sequence analysis determines the expected result of executing a generated sequence and the level of confidence in achieving that result. Automatic analysis by the computer may provide tests for dynamic loading, collisions, valid range of motion, and valid commanded velocities and accelerations. Local-site monitoring analyses the reports from the remote site to test for valid execution and system health. The local site will usually have much greater diagnostic capabilities than the remote site due to the greater human and computational resources available. Local-site telemetry provides the communication of command sequences to the remote site and the reception of status and data from the remote site.

Remote-site telemetry receives command sequences from the local site and sends status data to the local site. Command sequences are parsed at the remote site into individual commands for execution. Sequence control provides the transition to the next command in a command sequence on expected termination and transition to reflex action on a reflex monitor event. Real time control provides the closed-loop servo control of the remote-site mechanisms. The control is based on commands generated at the local site. Remote-site monitoring is responsible for the analysis of remote-site execution, providing information on whether to implement the transition the state of execution. Reflex is the ability to respond to monitored conditions. The most common reflex is to transit to the next command in a command sequence based on a monitor event which indicates that the previous command has been successfully completed. An equally important reflex is the ability to transit to a safety reflex action based on an unexpected monitor event.

A local-remote system architecture incorporating supervised autonomy concepts is shown by the block diagram in Fig. 4. This architecture is part of a work developed by the Jet Propulsion Laboratory (JPL), at California Institute of Technology (Caltech) [3]. That work presents the simulation the features and capabilities of a system providing supervised autonomy of a remote manipulator system through the implementation of an operational laboratory simulation system. The system employed specific sensors, such as a stereo camera, and two motion-cooperative robotic arm manipulators as actuators.

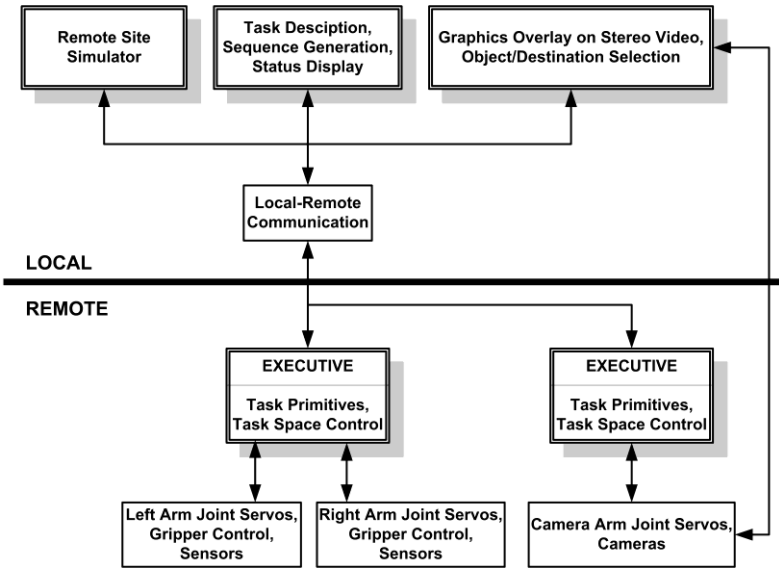

Fig. 4. Laboratory simulated local-remote system block diagram (P. G. Backes [3], p. 143)

In the system above the primary operator interface workstation provides interactive task description, sequence generation, and status display. A graphics workstation provides a stereo graphics overlay on stereo video as well as an interactive designation of objects or destinations. The remote-site simulator simulates remote-site execution with execution status displayed on the primary workstation and motion displayed on the graphics workstation. The remote site provides two control systems, one for independent, coordinated, or cooperative control of the two manipulators, and one for control of a third manipulator for positioning a suite of four cameras. The system's executive module provides communication with the local site and initiates task commands as specified by the local site. Task primitives provide joint and task space control and monitoring of single

or dual cooperating manipulators.

The remote-site system design of a space telerobotics system has more constraints imposed on it than the local-site system. A primary remote-site constraint is flight qualification of the software. This creates the need for fixed flight software, which has been validated before flight (or modified, validates and uplinked). Fixed flight software precludes custom optimized programs for each mission task. Rather, the fixed flight software must provide sufficient functionality to complete both expected and unexpected mission tasks.

The solution provided in that laboratory system [3] is a family of parameterizable task called primitives, each of them with a general functionality for a class of manipulation tasks. Separate commands provide other needed capability such as database update, status request, and execution interruption. Task execution primitives are self-contained programs, which provide the manipulator control capability with behaviors as specified by an input parameter set. A natural interface between the local and remote-site systems is then the parameter lists for the various task primitives.

The executive provides functionality similar to that of a spacecraft command and data subsystem. It receives commands from the local site, parses the commands to determine command types, and initiates execution of the commands by executing task primitives or other commands with the parameterization given in the command data sets. The executive also returns system state information to the local site. The interface commands that can be sent to the remote site by the local site include database, status, and execution commands. The database command has parameters specifying the arm and database datatype followed by the specific database parameters. The task primitives along with the task primitive parameters when executing a task use the database parameters. The status command requests that the remote site return the state of the arm specified in the command. Finally, the execution command effectively starts the preset autonomous motion execution tasks of the arm, such as moving to touch or grasp commands.

The remote-site system design specifies the interface that the local site can use to control the remote manipulators. The local-site system is then designed to provide the remote-site capability to the operator. The task descriptions and sequence generations are provided by the User Macro Interface (UMI). The UMI abstracts away the details of the local-remote interface and provides the operator with more natural menus for specifying tasks and parameterization. The resulting inputs from the operator are converted to equivalent commands and parameterization to be communicated to the remote site. The operator has the option of running either the real remote-site robots or simulating the motion at the local site by sending commands to the remote-site simulator (which is physically located at the local site) and observing the results on the graphics display. The simulation mode is selected as a parameter in the UMI environment menu. The remote-site simulator runs identical control software as in the remote-site system and sends joint angle data to the UMI graphics displays. The UMI eventually specifies task primitives and their parameterization to the local-site executive to perform the specific tasks desired by the operator. The operator may save a specific parameterization of a task as a task command for later utilization. The status of the remote-site system is updated on the local-site operator control station monitor whenever a system status or command result is returned from the remote site.

When handling with telerobots that perform controlled autonomous motion, it is important to consider the effects caused by relative and absolute motion commands. A relative motion command produces the same relative motion from the starting point as when the motion was told to begin, even though the absolute starting point changes. In other words, the motion is always relative to its starting reference, not taking into consideration whether this initial point has been changed. This is useful for tasks being executed relatively to their environment. Absolute motion commands have an absolute position destination independent of where the motion has started. This is useful for moving to an absolute position before beginning a relative motion command.

3.2. Command sequence controlling

Sequence control is responsible for the management of the transition between commands in a sequence or transition to a reflex action. Two important parts of sequence control are: runtime binding, executed before each command, and termination condition testing, executed at the end of a command. Runtime binding actually binds parameters to a task command just before its execution is initiated. Parameters bound at runtime may not be known at the time the command is built. Some examples of runtime binding include binding the current safety parameters, speed factor, and the reporting period to the parameter list. Each command in a task sequence completes due to satisfaction of a termination condition (including safety conditions). If the termination condition is one of the acceptable termination conditions specified in the command, then the next command of the sequence is issued. If not, then a safety reflex action is initiated, and a new command sequence must be sent.

Transition between commands in a sequence can occur either at the local or the remote site, but transitioning to a reflex action should be done autonomously at the remote site. For sequence control at the local site, a delay at least as long as the round-trip time delay will occur between execution of each command in a sequence, because the local site must then receive the remote-site status indicating that the command has been terminated successfully before sending the next command in the sequence. This situation implies the need of some round-trip time delay measurement equipment, based on data transmission estimatives and network latency real time information. In case of the local site receives a timeout status from the last command sent, it means that a middle-way problem has been detected, such as a possible loss of data or a breakdown communication with the remote site.

4. COLLISION PREVENTION TECHNIQUES FOR SPACE MANIPULATORS

Space manipulators, along with space telerobots, are designed and developed to operate in environments that differ from the usual Earth-grounded ones. When these telerobots are not performing tasks on other planet (or other Astros) surfaces, they usually find themselves in zero-g

environments, performing operations suitable to those gravity/friction/drag-free locations. Such operations also require special control algorithms developed for space robotics. Regarding those differences between terrestrial and space robotics, there are three main issues: unconstrained motion, stability during the contact transition, and force controlled manipulation of the environment.

Unsuccessful autonomous unmanned/telerobotics mission failures due to spacecraft impacts have been reported, such as the one experienced by the DART space vehicle [6]. DART was a spacecraft project which focused its mission on the testing of autonomous rendezvous and docking technology application. The spaceship was expected to rendezvous with MULBCOM satellite, launched on May, 1999, and grasps this obsolete satellite. But less than 11 hours into the mission, the DART collided with MULBCOM. This fact enhances the risky features associated to unexpected collisions.

If a space robot is unattached to its environment, it is considered that the robot is free-flying robot or a robot having a moving-base, forming two independent systems. In this case, the main preoccupations of the designers include: path planning, obstacle avoidance, and rendezvous and docking. Force control is not pertinent, because any forces exerted between the robot and its environment will tend to repel each of them away from the other. Therefore, if force control is to be applied, the robot should attach itself to the environment, making a continuous kinematic chain. The attachment of the robot to the environment is typically achieved through slow docking followed by base attachment or grasping by one arm or a multi-arm system. Once the robot and its environment are coupled, all three problems, robot collision-free motion, contact transition, and force control, are important. Collision-free motion is often more difficult when including the constraint of the attached base. However, the attachment does allow the control of interaction forces, in the form of impact control and accurate force trajectory following on the contacted surfaces [7].

The author divides the research of obstacle avoidance to reach the goal location into two classes of methods: global and local. Global methods rely on the description of the obstacles in the configuration space of a manipulator. Local methods rely on the description of the obstacles and the manipulator in the Cartesian workspace, referring to the local working envelope of both systems.

Global methods require two main problems to be addressed. First, all the obstacles of the environment must be mapped into the configuration space of the manipulator. Second, a path through the configuration space must be found for the point representing the manipulator. Two techniques are used to generate those paths: geometric searches and artificial forces. The geometric search technique relies on an exhaustive search of the unoccupied configuration space for a continuous path from the start point to the goal point. The artificial force technique surrounds the configuration space obstacles with repulsive potential energy functions, and places the goal point at a global energy minimum. The point in the configuration space representing the manipulator is acted on by a force equal to the negative gradient of this potential field, and driven away from obstacles and to the minimum. In the real flight software, the real time control should be based in those potential functions, calculating the current manipulator positions and making them deviate from the gradient peaks, which means the environment obstacles.

Global methods have several disadvantages. The algorithms necessary for global methods are computationally intensive. Thus, they are suited only for off-line path planning and cannot be used for real time collision avoidance. An immediate consequence is that global algorithms are difficult to use for collision avoidance in dynamic environments, where the obstacles are moving in time. Also, when using global algorithms it becomes very difficult to describe complicated motion planning tasks such as those performed by two manipulators moving cooperatively.

The local methods are possible alternatives to global methods. Local methods also employ the use of artificial forces. However, unlike configuration space forces, local forces are expressed in the Cartesian workspace of the manipulator. Object collisions are prevented by surrounding them with repulsive potential functions, and the goal point is surrounded by an attractive well, as shown in Fig. 5. These potentials are added to form a composite potential which imparts forces on a model of the manipulator in Cartesian space. Torques equivalent to these forces cause the motion of the real manipulator. Those models referred here are computed using mathematical dynamic modeling of specified robot manipulators [8].

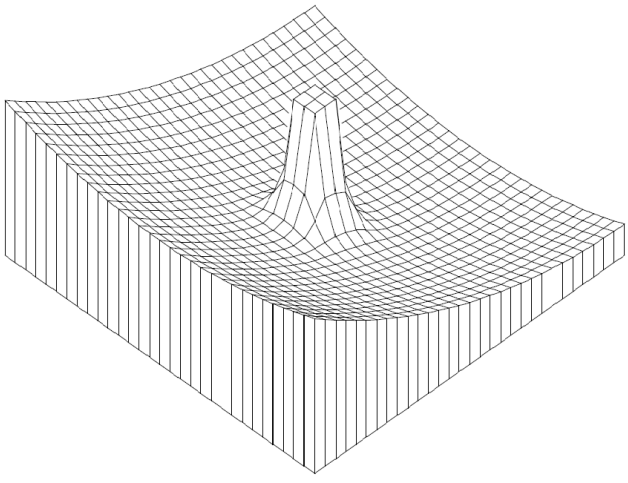

Fig. 5. Repulsive potential added to an attractive well (R. Volpe [7], p. 178)

The main advantage of local techniques is that they are less computationally demanding than the global ones, permitting their use in real-time control. Further, they provide the necessary framework to deal with changing environments and real time collision avoidance. When used

with a teleoperated manipulator, local artificial forces also provide low-level collision avoidance, while human operators perform high-level path planning of the manipulator.

5. REAL TIME DISTRIBUTED AND PARALLEL SOFTWARE SYSTEMS

The need for computer simulation of real systems before implementing tasks associated with the project development is very important and is considered strictly necessary. In this sense, the easy access to computer facilities always plays a fundamental role in simulating real time systems. The ever-increasing advancement of computer-based technology used to be isolated systems, are now connected together to form a complex "system of systems" [9]. In this reference the author explains that simulation does not replace the fundamental needs for "live" training activities, but it allows them to focus on high value training evolutions by implementing preparatory activities to be undertaken, in advance, in virtual environments.

A computational model as defined here refers to the computer architectures that represent the space telerobot system dynamics as it would be in the remotely space environment. The associated computer simulations are processes of reproducing the behavior of some real system (space system for example) approached by mathematical models. In this regard, the parallel/distributed simulation technologies enable a simulation program to be executed on parallel/distributed computer systems. Such systems are also usually known as clusters of workstations, namely, systems composed of multiple interconnected computers, or multiprocessor systems, that is, many processors interconnected in the same computer allowing program parallel executions. Fujimoto [10] comments the primarily four principal benefits of executing a simulation program across multiple computers: reduced execution time, geographical distribution of machines and users, integrating simulators that execute on machines from different manufacturers and fault tolerance.

The scenario concerning telerobotics supervised autonomy, as described in the section 3, may require exactly the parallel/distributed real time simulation purposes introduced here. In other words, if there exists a more powerful computational resource at ground stations (local site) to simulate pre-planned generated commands, it becomes strictly necessary to simulate those commands before sending them to the space telerobot (remote site). Thus, using a real time distributed/parallel software system to accomplish those types of tasks would be a reliable solution to measure the space telerobot systems effectiveness, safeness and fault-tolerances. Hence, once the system pre-requisites are defined, it becomes necessary to define a distributed virtual architecture that allows the real scenario representation by simulating it in a distributed environment. The three main existing architectures known in the literature are: Distributed Interactive Simulation (DIS), High-Level Architecture (HLA) and Test and Training Enabling Architecture (TENA) [9].

A very simple, stable, reliable and validated architecture available today is the DIS. The DIS architecture is comprised by a communication standard that provides a method of communicating entity state and other information by means of message packets called Protocols Data Units (PDUs). The DIS is nowadays considered as a fully mature simulator/simulation communication technology. It employs a 3D geocentric coordinate system and a standard set of dead reckoning algorithms to reduce the required bandwidth. The PDUs are all standardized and enumerated in a comprehensive way so as to comprise entities, sensors, communication devices, environmental descriptors, collision detectors and other attributes. It is observed that DIS is mainly focused in the communication methods between entities or software systems, which correspond to the data flow bus analysis presented by the laboratory simulated local-remote system scenario referenced in this paper at section 3. DIS is also not so complicated to install over an existing infrastructure, requiring only a network of suitable necessary bandwidth and latency. If the simulation could run effectively in a DIS virtual environment, then it will be simpler to transport the validated system to a real test platform including real time operating systems containing distributed/parallel shared memory applications, physical sensors and links susceptible to noised measurements, besides other features inherent of real-working engineering systems.

6. DISCUSSION

Software systems incorporating technologies that involve real time modeling, simulation and control represent today a newly upcoming effort from software engineers, whose goals are mainly toward to the development of robust simulation and control frameworks. Regarding the Brazilian updated space mission projects, it could be addressed to the SARA Space Retrievable Orbital Platform [11]. SARA is an acronym for SAtellite for Re-entry in Atmosphere, and shall include a docking port for small satellite coupling. This feature is necessary for the SARA objective accomplishment of space rendezvous and docking maneuvers. The idea is to provide the platform with the capability of on-orbit maintenance and on-orbit experiment replacement [12]. Some researches to introduce real time computational simulation and control planning into SARA's onboard computer simulators are discussed in [13]. Those studies aim to meet the software requirements that would allow the successful achievement of the spacecraft guidance, navigation and control to perform the rendezvous and docking maneuvers autonomously. Again, it shall be taken into consideration techniques employing real time supervised autonomy, collision prevention, and previously simulations which shall use distributed simulation software systems.

7. CONCLUSION

Technological approaches on the planning of complex space mission designs are always concerned on how reliable would be the mission teleoperated systems. This work has described how a teleoperated or telerobot system works with emphasis on the computing subsystems. Special emphasis has been put on software systems available that compose these subsystems. The paper has also presented the supervised autonomy concept, one of the most useful space telerobotics automatic control policies. A discussion about risky space maneuvers has been addressed here by detailing

the collision prevention techniques used onboard telerobots. All those subjects have then been related to the recent technologies involving real time simulation and control, usually performed in such a distributed or parallel virtual environment. Some architecture available to recreate complex real time system scenarios has also been commented. The issues presented by this work aim future Brazilian space mission project applications, such as the SARA spacecraft, which intends to demonstrate near-term technology implementation.

ACKNOWLEDGMENTS

The authors would like to acknowledge the Brazilian Institute for Space Researches (INPE) and the National Counsel for Scientific and Technological Development (CNPq) for supporting this work development, and the cooperation of Paulo Moraes Jr. along with the Aeronautics and Space Institute (IAE).

REFERENCES

[1] C. F. Ruoff, "Overview of Space Telerobotics", Teleoperation and Robotics in Space, Progress in Astronautics and Aeronautics, Vol. 161, pp. 3-21, USA, 1994.

[2] C. S. Hartley, and R. Pulliam, "Use of Heads-up Displays, Speech Recognition, and Speech Synthesis in Controlling a Remotely Piloted Space Vehicle", IEEE AES Magazine, pp. 18-26, 1988.

[3] P. G. Backes, "Supervised Autonomy for Space Telerobotics", Teleoperation and Robotics in Space, Progress in Astronautics and Aeronautics, Vol. 161, pp. 139-158, USA, 1994.

[4] D. M. Harland and R. Lorenz, "Space Systems Failures: Disasters and Rescues of Satellites, Rockets and Space Probes", Springer / Praxis Publishing, UK, 2005.

[5] J. L. Lions, "Ariane 5 Flight 501 Failure Report by the Inquiry Board", France, July 1996. Available at <http://sunnyday.mit.edu/accidents/Ariane5accidentreport.html> - Access on 04/02/2008.

[6] Overview of the DART Mishap Investigation Results for Public Release, April 2005. Available at <http://www.nasa.gov/mission_pages/dart/main> - Access on 04/07/2008.

[7] R. Volpe, "Techniques for Collision Prevention, Impact Stability, and Force Control by Space Manipulators", Teleoperation and Robotics in Space, Progress in Astronautics and Aeronautics, Vol. 161, pp. 175-212, USA, 1994.

[8] J. J. Craig, "Introduction to Robotics", Mechanics and Control, 2nd Ed., Addison-Wesley Publishing Company, 1989.

[9] D. Mc Farlane, "Distributed Simulation Guide", Australian Defence Simulation Office, Department of Defence, Australia, 2004.

[10] R. M. Fujimoto, "Parallel and Distributed Simulation Systems", John Wiley & Sons, Inc., USA, 2000.

[11] P. Moraes Jr., "Design Aspects of the Recoverable Orbital Platform SARA", 8th Chilean Congress of Mechanical Engineering, Chile, October 1998.

[12] N. Seito, I. M. da Fonseca, and P. Moraes Jr., "SARA Orbital Autonomous Rendezvous and Docking. A Scientific and Technological Challenge", 6th Brazilian Conference on Dynamics, Control and their Applications, São José do Rio Preto, Brazil, May 2007.

[13] J. Gentina, I.M. da Fonseca, and P. Moraes Jr., "Technology Overview on the Development of a Computational Module for Space Docking Maneuvers", not published in the V National Congress of Mechanical Engineering, Salvador, Brazil, August 2008.