A REAL-TIME SIMULATION ARCHITECTURE FOR MULTI-ARM SPACE ROBOTS BASED ON RAPID PROTOTYPING

*Mingming Wang[1], Jianjun Luo[2], Lijun Zong[3], Ulrich Walter[4]

[1]Science and Technology on Aerospace Flight Dynamics Laboratory, Youyixilu 127, 710072, Xi'an, P. R. China, E-mail: mwang@nwpu.edu.cn

[2]Science and Technology on Aerospace Flight Dynamics Laboratory, Youyixilu 127, 710072, Xi'an, P. R. China, E-mail: jjluo@nwpu.edu.cn

[3]Science and Technology on Aerospace Flight Dynamics Laboratory, Youyixilu 127, 710072, Xi'an, P. R. China, E-mail: zong0428@126.com

[4]Institute of Astronautics, Boltzmannstr. 15, 85748, Garching, Germany, E-mail: walter@tum.de

ABSTRACT

This paper presents a real-time simulation architecture for multi-arm space robots based on rapid prototyping. The objective of this paper is to make the simulation architecture open for the research of on-orbit autonomous mission execution and on-ground collaborative tele-operation. Additionally, the simulation architecture provides the monitor and operator on-ground an intuitive view of space robotic operations in a wide set of scenarios. The mission profile and background of space robotic operations are firstly recalled. The concept of rapid prototyping and its developing procedure are introduced which will be employed in the simulation system design. Within this context, a close look into the Sim-SOMAR system about overall design and simulation environment are described. Next, the detailed characters of real-time communication mechanisms about real-time simulation system are exhibited. The components of Sim-SOMAR (Simulation of Space Operations with Multi-Arm Robots) system, which includes multi-body dynamics, autonomous mission management, path & trajectory planning, controller design and telemetry & telecommand subsystems are introduced. Additionally, a user-friendly virtual reality (VR) subsystem for coexistence of working operators and space robot is developed, which is composed of haptic input devices, VR helmet and Head-Up display (HUD). Well-designed simulation system architecture makes the Hardware-in-loop (HIL) simulation possible and can be extended easily in the future.

1 INTRODUCTION

Space debris is now considered as a serious issue for the future space operation. Using space robots to remove these debris is one of the most reasonable and feasible methods [1]. Besides, the increasing demands of satellite maintenance, on-orbit assembly and re-supply also call for applications of space robot to perform tasks in the particular harsh space environment. Therefore, the applications of space robot have been attracted plenty of attentions and some investigations have been conducted in the field of space robot, such as "Robot Technology Experiment (ROTEX)" [2], "Ranger" [3], "Engineering Test Satellite VII (ETS-VII)" [4], "Orbital Express (OE)" [5], and series of ongoing space robot projects by worldwide space agencies. Before an artificial space robot launches into orbit, a series of system performance analysis and validation have to be implemented to ensure the success of the ongoing space project. Hence, a real-time distributed implementation that can emulate various scenarios of space missions is highly necessary. This motivates us to build a Sim-SOMAR (Simulation of Space Operations with Multi-Arm Robots) system based on rapid prototyping in Science and Technology on Aerospace Flight Dynamics Laboratory (AFDL), Northwestern Polytechnical University. A schematic diagram of multi-arm space robot with an un-cooperative target is shown in Fig. 1.

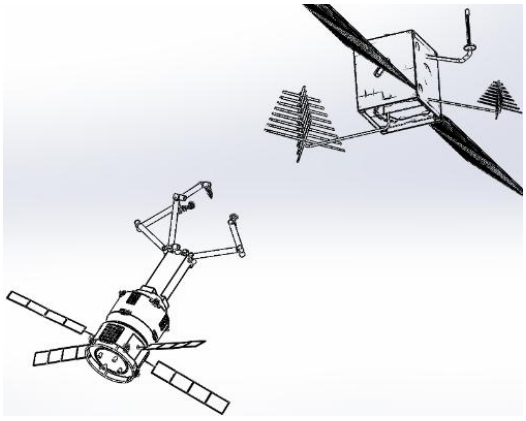

Figure 1: Multi-arm space robotic mission.

In order to validate the performance and effectiveness of the designed algorithms like muti-

body dynamics, path planning, motion controller as well as human-in-the-loop & hardware-in-the-loop simulation, some simulators for humanoid, aircraft and unmanned autonomous vehicle (UAV) have been proposed in the literature based on common object request broker architecture (CORBA) [6], date distribution service (DDS) [7] and rapid prototyping (RP) [8]. RP techniques allow to implement and validate the designed strategies during the development process. The designer can work within the same environment from the requirement analysis to the system design and implementation phase. The design efficiency and development cost are improved accordingly, which also motivate us to employ RP techniques in AFDL to construct a real-time distributed simulator for multi-arm space robotic missions.

The objective of this paper is to make the real-time simulation architecture open for the research of on-orbit autonomous mission execution and on-ground collaborative tele-operation in a wide set of scenarios. Some considerations [9] taken into account for selection of the appropriate implementation approach are

1. Easy operation both at novel level and expert level;
2. Simulation system should be flexible enough to run various cases and scenarios;
3. The interaction machine-user interface should be highly intuitive;
4. Simulation system should be easily maintainable and expandable.

The rest of the paper is organized as follows. Section II introduces the space robotic mission profile. Section III gives the overall Sim-SOMAR design and presents the simulation environments, RP techniques and virtual reality in detail. Section IV illustrates the design of Sim-SOMAR system, which is composed of multi-body dynamics, autonomous mission management (AMM), path & trajectory planning, and controller design. The proposed RP-based real-time simulation architecture can be utilized to analyze and evaluate the performance and feasibility of a certain multi-arm space robot and provide intuitive view for the operator to complete various space missions.

2 MISSION PROFILE

A typical space mission is conducted in a series of operations. In order to perform the space missions using space robot, such as on-orbit assembly or capturing a tumbling target, the motions of space robot can be divided into six phases as shown in Fig. 2 which will be illustrated in the following.

Rendezvous During this phase, the servicer satellite has completed its orbit phasing and has entered a nearby orbit of the target. Actions of the servicer in this phase include acquiring and updating the orbit knowledge of the target, synchronizing the mission time-line, and regulating the necessary pose and velocity. According to the possible encountering anomalies, such as losing visual contact with target or incorrect approach rates, the servicer can transited to an error-recovery mode.

Formation Before attempting any contact operations with the target, the servicer has to obtain the information of the target and ensure the safety during the space operation. This can be achieved by the formation flying. During this phase, the servicer is in very close proximity of the target such that the target is within the reach of the servicer robotic arm.

Approach The approach phase starts from an observation distance and brings the manipulator mounted on the servicer to an optimal grasping pose. After that, a tracking phase will be proceeded. The approach phase moves space robot end-effector from an initial position to a final position under certain constraints. The final position of the approach phase will be the initial position of the tracking phase.

Tracking When space robot needs to capture or observe a certain point of a tumbling target, a tracking mission will be required. This tracking phase aims to minimize the residual relative velocity between the target interested point and the space robot end-effector. The duration of tracking phase depends on the tracking controller performance, such that the end-effector can have sufficient time to regulate its pose and ensure the success of tracking the target interested point. At the end of tracking phase, neither the end-effector nor the robot joints are at rest. Their states will be the initial value of the next phase.

Stabilization Once the relative distance between space robot end-effector and target interested point reduces to zero and the gripper on the end-effector is closed, in order to retain the stable attitude of the target, a stabilization phase must be carried out to decrease the angular velocity of the target to zero and be ready for further space operations. In the stabilization phase, by exerting external forces and torques or adjusting the internal torques of robot joints simultaneously, the relative motion between the servicer and the target will be brought to zero.

Operation Once the space robot completes the stabilization mission and makes the servicer dock with the target, further operations such as on-obit maintenance, assembly as well as unit exchange can be performed to implement various the on-orbit servicing missions.

In fact, besides above general uses of space robot, there are still some other potential applications of space robot. For example, as a result of dynamic

coupling between spacecraft and space manipulator, it can be utilized to regulate spacecraft attitude for decreasing on-board fuel consumption. But these are not of this paper interests. According to the external force and torque exerted on the spacecraft, the servicer satellite will be operated in three flying modes. If there are no forces and torques exert on the servicer, it is termed free-floating mode. When only torque is applied to the servicer, it is called free-flying mode. Another mode is auto-flying mode. Before-mentioned three kinds of space phases can be performed under these three flying modes. The flying modes switch can be controlled by the operator in terms of the particular requirements.

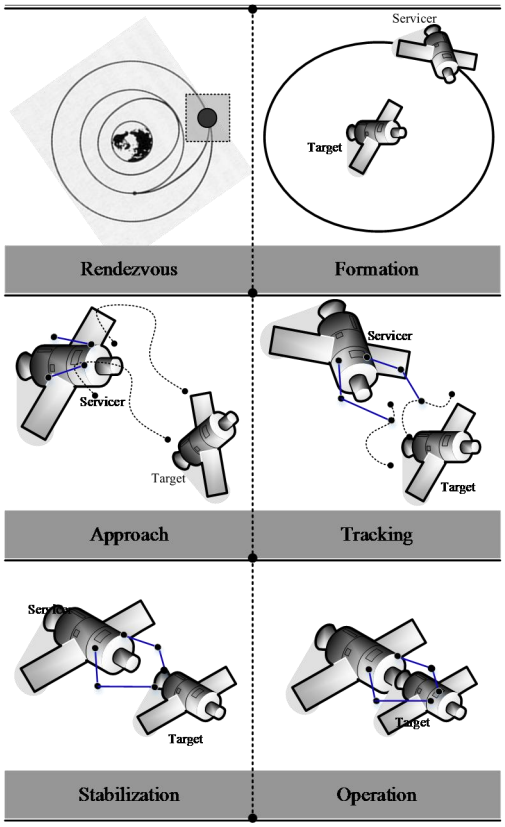

Figure 2: Mission Profile of Space Robot.

3 HIL ARCHITECTURE

This section illustrates the related techniques and simulation environment of designing of the Sim-SOMAR system.

3.1 Sim-SOMAR Design

The Sim-SOMAR system is designed to execute autonomous mission and assist the operator with intuitive view, and validate the feasibility and the performance of the on-going space robot project. Accordingly, the following subsystems will be included in the Sim-SOMAR system. A schematic diagram of Sim-SOMAR system is shown in Fig. 3.

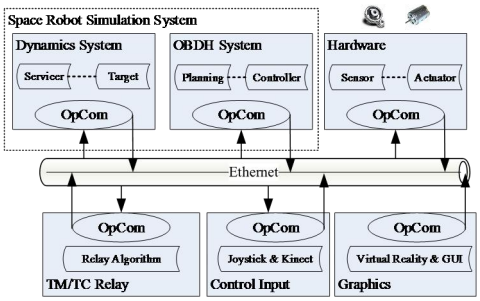

Figure 3: Sim-SOMAR Components.

Operator Console: It is an alias of Mission Control Centre (MCC). The operator will send a series of commands from MCC uplink to the service satellite via the Geostationary (GEO) relay satellite. The communication relay will be simulated by adding an individual node in the simulation system to relay the information. MCC will also receive the telemetry (TM) information collected from the states of space robot & target satellite and display them on a big screen to help the operator enhance the awareness and understand of the real-time situation in space. The operator can also control the space robot in real-time by joystick or 3D space mouse. A Virtual Reality (VR) system with related helmet for providing intuitive view is also running when the operation is proceeding. These will be introduced in detail in the Section 3.4.

TM/TC: This subsystem is responsible for the signal relay between MCC and GEO relay satellite. It transmits the tele-commands (TC) from MCC to the GEO relay satellite and the telemetry information from servicer to the MCC. The information dissemination between various nodes is based on IEEE1394.

Space Robot Simulation: This system contains two subsystems, one is dynamics of servicer and target, and the other is On-Board Data Handling (OBDH) subsystem. For detailed description of space robot simulation system design will be presented in Section 4.

Hardware-in-loop (HIL): This system can include different types of hardware, such as sensors, actuators, etc. Since the expandability and scalability of the simulation system, the proposed simulation architecture is easy to replace the software by particular hardware.

3.2 Rapid Prototyping

The concept of rapid prototyping (also named V-model) originated from 1980s and became available in the field of 3D printing or additive layer manufacturing. It also spread its application

to the software development. At the core of RP technology is the integration of the implementation and validation for the designed strategies during the development process. The designer can work within the same environment from the requirement analysis to the system design and implementation phase. As shown in Fig. 4, the left side identifies steps that lead to code generation, including requirements analysis, system specification, detailed software design, and coding. The right side focuses on the verification and validation of steps cited on the left side, including software integration and system integration.

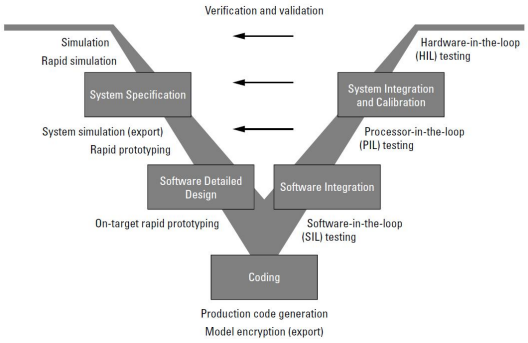

Figure 4: Rapid Prototyping.

Depending on the application and its role in the process, we focus on all the steps expressed in the V-model and repeat steps at several stages of the V-model. Code generation technology and related products provide tooling that we can apply to the V-model for system development. For more information about the application of RP techniques and code generation, one can refer to the literature [9].

3.3 Simulation Environment

One of the first and major implementation solutions adopted was to build Sim-SOMAR on top of Matlab environment (see Fig. 5). This environment integrates sufficient properties to cope with the aforementioned requirements, assuring high flexibility and availability of existed functionality.

Normally, modern engineering design starts from the application of Computer Aided Design (CAD) system, such as CATIA or 3ds Max. CAD can assist the designer in the creation, modification, analysis, or optimization of a design. On the one hand, it provides the designer an efficient and intuitive tool to improve the quality of the design and increase the productivity. On the other hand, the objects designed by CAD will be as a basic input of VR. In this paper, Unity 3D is chosen to represent 3 dimensional (3D) interactive vector graphics. In order to increase the interactivity and immersion of the operator, C#/JavaScript is integrated into Unity to deal with the interactive event response issue. The transform between CAD and Unity can be found from the relevant literature.

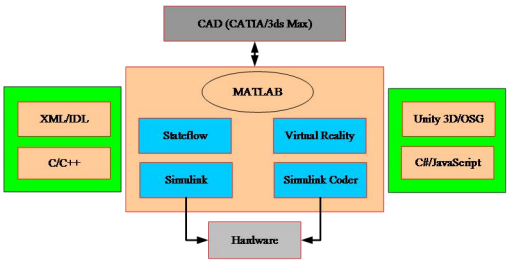

Figure 5: Simulation Environment.

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. Its main design goals are simplicity, generality and usability over the Internet. In our design, XML is utilized to describe the configurations and initial states of the Sim-SOMAR system.

As a major programming language, C/C++ programming language can establish connections among various applications. That's why it is also employed here to deal with the different interfaces of relevant applications involved in Sim-SOMAR. Before simulation starts, it proceeds the XML file to extract the configuration and initial states of the simulation system.

At the core of Sim-SOMAR system is Matlab/Simulink/Stateflow/Simulink Coder, where Simulink is a data flow graphical programming environment for modeling and simulating multi-domain dynamic systems. A number of hardware and software products are available for use with Simulink. Stateflow, integrated in Simulink, provides a control logic tool to model reactive systems via state machines and flow charts. Coupled with Simulink Coder, Simulink model can automatically generate C/C++ source code for distributed real-time simulation.

3.4 Virtual Reality

In order to provide an immersion and intuitive feeling for the operator, development of a virtual reality system is significantly necessary. As explained in Section 3.3, the design of space robot relies on CAD, which can transform the original CAD model into Unity 3D. With adding specific C#/JavaScript applets, under the Simulink environment, a complete VR system can be established. For the control input of operator, joystick and 3D mouse are employed. When a space robot task is processing, it is important for the VR to supply the full scene of space robot and the relative

scene between the end-effector and target. The relative scene will offer a viewpoint on top of end-effector to target satellite. Therefore, a Head-Up Display (HUD) [10] with full of relative information and explicit vision of target satellite is built to assist the operator in the tele-operation tasks.

4 SPACE ROBOT SIMULATION SYSTEM

This section illustrates the structure and composition of the space robot simulation system. It contains the following subsystems, which are multi-body dynamics, autonomous mission management (AMM), path & trajectory planning, and motion controller.

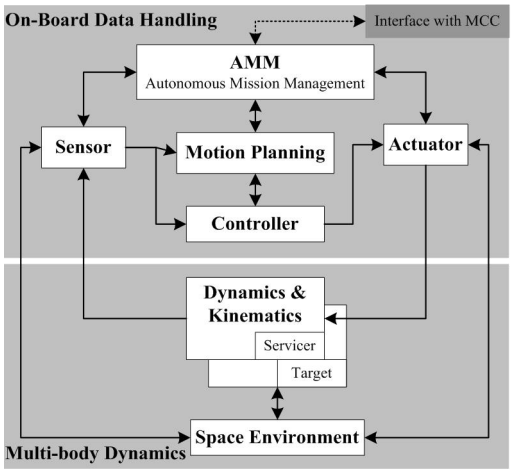

Figure 6: Implementation of Sim-SOMAR System.

4.1 Multi-Body Dynamics

Multi-body dynamics modeling can be divided into two categories: operational-space dynamics and joint-space dynamics. Since the singularities in operational-space are unpredictable, especially when space robot involved in free-floating mode, thus joint-space dynamics modeling is adopted. Many investigations have been conducted in the filed of space robot dynamics since the earliest work more than three decades ago. In this paper, a modified composite-rigid-body-algorithm (CRBA) based on graph theory and spatial notation proposed in is adopted. Suppose k manipulators with in total n degree-of-freedom (DOF) are installed on the servicer satellite, the general dynamics equation can be described by the following expression:

$$\begin{bmatrix} \mathbf{H}_b & \mathbf{H}_{bm} \\ \mathbf{H}_{bm}^T & \mathbf{H}_m \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{x}}_b \\ \dot{\boldsymbol{\mu}} \end{bmatrix} + \begin{bmatrix} \mathbf{c}_b \\ \mathbf{c}_m \end{bmatrix} = \begin{bmatrix} \mathbf{f}_b \\ \boldsymbol{\tau} \end{bmatrix} + \sum_k \mathbf{J}_e^{kT} \mathbf{f}_e^k$$

where $\mathbf{H}_b$, $\mathbf{H}_{bm}$ and $\mathbf{H}_m$ are the base inertia matrix, dynamic coupling matrix and manipulator's inertia matrix, respectively, $\mathbf{q} = [\mathbf{x}_b; \boldsymbol{\mu}]^T$ represents the velocity of system states, while $\mathbf{x}_b = [\mathbf{r}_{C_0}; \boldsymbol{\omega}_0]^T$ is the translational and angular velocity vector of the spacecraft, $\boldsymbol{\mu} \in \mathbb{R}^{n \times 1}$ is the joint velocity vector of space manipulator. $\mathbf{c}_b$ and $\mathbf{c}_m$ are the Coriolis and centrifugal force vectors of the base and manipulator, respectively, $\boldsymbol{\tau} \in \mathbb{R}^{n \times 1}$ are the torques of the joint actuators, $\mathbf{f}_b \in \mathbb{R}^{6 \times 1}$ and $\mathbf{f}_e^k \in \mathbb{R}^{6 \times 1}$ are the generalized forces applied on the base and end-effector k, respectively. $\mathbf{J}_e^k \in \mathbb{R}^{6 \times n}$ is the Jacobian matrix of k^{th} end-effector.

4.2 Autonomous Mission Management

In the space robot simulation system, the space tasks can be conducted in manual or autonomous operational modes. In the manual mode, an operator sends commands or utilizes joystick or other control instruments to perform the space mission. In the autonomous mode, the space mission is conducted fully autonomously with a minimum number of interventions from the operator.

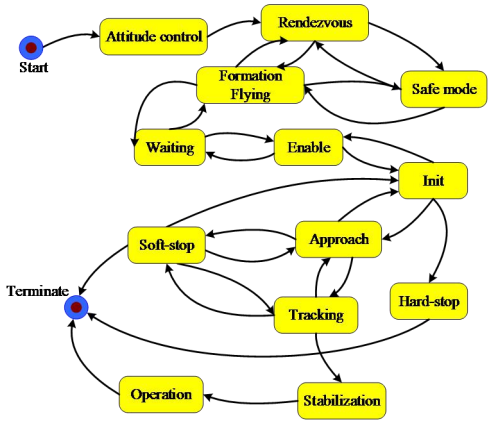

Figure 7: Implementation of Sim-SOMAR System.

AMM is fully at the core of the On-Board Data Handling (OBDH) system of the servicer satellite. The fully implementation of AMM relies on the preloaded space missions, which are designed and encoded with a hierarchical Finite State Machine (FSM) engine such as Matlab/Stateflow. The concept of the hierarchical FSM allows a high-level FSM to activate or invoke a lower-level FSM. This supplies the capability to implement a hierarchical decomposition of a high-level task into a sequence of lower-level tasks. As shown in Fig. 7, various space missions are then considered as different states which can be switched from one to another when tele-commands sent by the operator are triggered or specific conditions are met. These specific conditions include the lighting condition, communication condition, thermal condition, or the

tele-commands sent by operators, etc.

The role of the operator is to initialize the space operations by submitting high-level command, conduct space mission manually in real-time by using specific instruments, and monitor the process of the servicer performing tasks. In case of emergency, the operator could send a halt or abort command with highest priority to the autonomy engine.

4.3 Path &Trajectory Planning

This subsystem receives commands from MCC and generates an appropriate path to accomplish the required task. Normally, for the multi-arm point to point approach task, an inverse kinematics algorithm must be included to find the optimal configuration of space robot in joint-space. Then different curves (cubic, quintic, B-Spline, etc.) can be implemented to fulfill the constraints of initial and terminal points. For tracking task, an inverse kinematics algorithm at velocity level has to be employed to follow a pre-defined path or the continuous input of the operator.

When a tracking task is performed, singularities maybe occur since the rank degradation of Jacobian matrix, which induce infinite velocity in joint-space. This phenomenon is high undesirable, hence, singularity detection and avoidance algorithms are attached to the path & trajectory planning. Another issue of concern is collision avoidance. When operating the space robot, it is possible not only to impact the target satellite, but also collide with the robot itself or the servicer satellite. These call for the collision detection and avoidance algorithm, which are also integrated into space robot simulation system.

According to above constraints, the general nonlinear optimization problem can be formulated as follows:

$$
\begin{aligned}
&\min_{f t_f ; \mathbf{q}(t) g} \; \psi(\mathbf{q}(t); \dot{\mathbf{q}}(t); t_f) \\
&\text{subject to} \\
&\begin{cases}
\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}; \dot{\mathbf{q}}) = \tau \\
\mathbf{h}(t_f ; \mathbf{q}(t)) \le 0 \\
\mathbf{g}(t_f ; \mathbf{q}(t)) = 0 \\
\mathbf{q}(t_0) = \mathbf{q}_{ini}; \dot{\mathbf{q}}(t_0) = \mathbf{0}
\end{cases}
\end{aligned}
$$

For $t_0 \le t \le t_f$ and t_f is the finial time. $\psi(\mathbf{q}(t); \dot{\mathbf{q}}(t); t_f)$ is a predefined cost function. $\mathbf{h}(t_f ; \mathbf{q}(t))$ are the inequality constraints, such as input/output constraints, collision and singularity constraints, etc. $\mathbf{g}(t_f ; \mathbf{q}(t))$ are the equality constraints, e.g. end-effector finial state constraints.

4.4 Motion Controller

Controllers are designed and implemented for realization of planning problem by using actuators and sensors under control law. In fact, path & trajectory planning describes the commanding position and motion strategy of space manipulator, based on the planned trajectory, employing on-board devices to perform the commanding motion. In this paper, three control laws, which are Proportional Derivative (PD) controller, Computed Torque Controller (CTC) and a Nonlinear Model Predictive Controller (NMPC) with collision avoidance constraints are adopted [11].The traditional PD and CTC strategies are employed for comparison reasons.

Model Predictive Control (MPC) originated from the chemical process industries and gradually broadened its applications into other disciplines. Its fundamental idea is using a prediction model and numerical optimization methods to obtain a sequence of control inputs that minimizes a predefined cost function over a given time horizon, subjects to certain constraints. At each sample time, based on the new measurements and the current states, the prediction model and optimization are performed to determine the input over the control horizon. The first control input is applied and process is repeated at the next sampling instant in a "receding horizon" manner. From a theoretical viewpoint, the NMPC algorithm can be given as follows

$$
\begin{aligned}
&\mathbf{u} = \arg_{\mathbf{u}} \min \psi(k) \\
&\text{subject to} \\
&\begin{cases}
\mathbf{x}(k|k) = \mathbf{x}_0 \\
\mathbf{u}(k+j|k) = \mathbf{u}(k+N_c|k); \; j, N_c \\
\mathbf{x}(k+j+1|k) = \mathbf{f}_d(\mathbf{x}(k+j|k); \mathbf{u}(k+j|k)) \\
\mathbf{y}(k+j|k) = \mathbf{h}_d(\mathbf{x}(k+j|k); \mathbf{u}(k+j|k)) \\
\mathbf{y}_{min} \le \mathbf{y}(k+j|k) \le \mathbf{y}_{max} \\
\mathbf{u}_{min} \le \mathbf{u}(k+j|k) \le \mathbf{u}_{max} \\
\mathbf{h}(\mathbf{x}(k+j|k)) \le 0; \; \mathbf{g}(\mathbf{x}(k+j|k)) = 0
\end{cases}
\end{aligned}
$$

where $j \in [0; N_p - 1]$, N_p and N_c are the prediction and control horizon, respectively. $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{u}$ represent states, outputs and control inputs of the system, respectively. $\mathbf{x}_0$ is the initial condition vector. The notation $\mathbf{a}(i|j)$ indicates the value of vector $\mathbf{a}$ at the instant i predicted at the instant i predicted at the instant j. $\mathbf{f}_d$ and $\mathbf{h}_d$ are the discrete system prediction model and measurement model. The control inputs $\mathbf{u}$ and outputs $\mathbf{y}$ yield to the lower and upper bound, while $\mathbf{h}(\mathbf{x}(k+j|k))$ and $\mathbf{g}(\mathbf{x}(k+j|k))$ stand for other inequality and equality constraints. The cost function $\psi(k)$ is determined in terms of the predicted and the desired output of the system over the prediction horizon. The optimization issue must be solved at each sampling time k to obtain a sequence of optimal

control inputs over the control horizon N_c as $\{u^*(k|k); u^*(k+1|k); \cdots; u^*(k+N_c|k)\}$. A multi-parametric Quadratic Program (mp-QP) method is utilized in the space robot simulation system to implement the NMPC controller.

Once implementing the Sim-SOMAR system development and verifying the algorithms with RP techniques, an on-ground simulation facility like OOS-SIM [12] will be established in AFDL with integration of the SIM-SOMAR system developed in this paper.

5 CONCLUSION

A real-time distributed simulation architecture based on the rapid prototyping (RP) technique has been presented in this paper. The proposed simulation architecture is an integrated environment for design and analysis of space robot system in a series of space scenarios. The integration of RP techniques and the commercial environment Matlab/Simulink decrease the communication latency among various simulator's subsystems and speed up the construction of the Sim-SOMAR system. Moreover, the reusable simulator is easy to modify and replace by better modules, which provides expandability and scalability of this simulation architecture in the future. The Sim-SOMAR system has demonstrated in its prototype the feasibility and effectiveness of such a complex but integrated environment. Future improvements will include extension to a wide range of space scenarios and enhancement of usability and scalability.

Acknowledgement

The authors gratefully acknowledge the support of the RT-Lab System and the Global Crown Company (Beijing). The research is supported by "the Fundamental Research Funds for the Central Universities".

References

[1] NASA. *On-orbit satellite servicing study: Project report*. National Aeronautics and Space Administration. Goddard Space Flight Center, 2010.

[2] G. Hirzinger, B. Brunner, J. Dietrich, and J. Heindl. ROTEX-the first remotely controlled robot in space. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 2604–2611, 1994.

[3] B. Bon and H. Seraji. On-line collision avoidance for the Ranger telerobotic flight experiment. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, pages 2041–2048, 1996.

[4] N. Inaba and M. Oda. Autonomous satellite capture by a space robot: world first on-orbit experiment on a Japanese robot satellite ETS-VII. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, pages 1169–1174, 2000.

[5] Andrew Ogilvie, Justin Allport, Hannah Michael, and John Lymer. Autonomous Satellite Servicing Using the Orbital Express Demonstration Manipulator System. In *Proc. of the 9th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS'08)*, pages 25–29, 2008.

[6] H. Yussof, G. Capi, Y. Nasu, M. Yamano and M. Ohka. A CORBA-Based Control Architecture for Real-Time Teleoperation Tasks in a Developmental Humanoid Robot. *International Journal of Advanced Robotic Systems*, 8(2): 29-48, 2011.

[7] S. Zheng, J. He, J. Jin and J. Han. DDS Based High Fidelity Flight Simulator. In: *Information Engineering, 2009. ICIE 09. WASE International Conference on*, 1: 548-551, 2009.

[8] M. Lizarraga, V. Dobrokhodov, G. Elkaim, R. Curry, I. Kaminer. Simulink Based Hardware-in-the-Loop Simulator for Rapid Prototyping of UAV Control Algorithms. AIAA 2009-1843, 2009.

[9] P. Colmenarejo, F. Gandia, A.G. Casas, A. Tomassini and F. Ankersen. GNCDE: An Integrated GNC Development Environment. In: *Proceedings of the 6th International ESA Conference on Guidance, Navigation and Control Systems*, 2006.

[10] M. Wilde, S.C. Hannon and U. Walter. 2012. Evaluation of Head-Up Displays for teleoperated Rendezvous & Docking. In *Proceedings of the 2012 IEEE Aerospace Conference*, pages 1-14, 2012.

[11] M. Wang, U. Walter, J. Luo and W. Ma. A DDS Based Real-Time Simulation Architecture for Space Robotic Tele-Operation. In: *64th International Astronautical Conference (IAC2013)*, Beijing, China, 2013.

[12] J. Artigas, M. Stefano, W. Rackl, R. Lampariello, B. Brunner, W. Bertleff, R. Burger, O. Porges, A. Giordano, C. Borst and A. Schaeffer. The OOS-SIM: An On-ground Simulation Facility for On-Orbit Servicing Robotic Operations. In *Proceedings of the 2015 IEEE International Conference on Robotics and Automation*, pages 2854–2860, 2015.