

Locomotion of Multi Arm Robot for In-orbit Operations(MARIO) in lab Environment

Praveen Elavazhagan¹[0009-0007-6089-8704], Cameron Leslie¹[0009-0009-0152-2664], Saurabh Upadhyay¹[0000-0002-2030-1594], Gilbert Tang¹[0000-0002-5497-7269], and Leonard Felicetti¹[0000-0001-6830-4308]

Cranfield University, Faculty of Engineering and Applied Sciences, College Rd,
Wharley End, Bedford MK43 0AL, United Kingdom
Praveen.Elavazhagan.489@cranfield.ac.uk, Cameron.Leslie@cranfield.ac.uk,
g.tang@cranfield.ac.uk, Saurabh.Upadhyay@cranfield.ac.uk,
Leonard.Felicetti@cranfield.ac.uk
<https://www.cranfield.ac.uk/>

Abstract. This paper presents the control architecture and experimental validation of MARIO, a multi-arm robotic platform developed by Cranfield University for In-Orbit Servicing and Manufacturing (IOSM) task demonstration in a laboratory environment. MARIO comprises three ROS-enabled, 6-degree-of-freedom robotic arms with a 5 kg payload capacity, mounted on a pneumatic air-bearing platform that enables near-frictionless planar movement over an epoxy floor. Each arm is equipped with onboard cameras and end-effectors to perform locomotion, assembly, and manipulation tasks with high precision. The developed ROS2-based control framework integrates a multi-layered architecture, combining a collision-free trajectory planner, synchronized multi-arm coordination, and hardware-level execution for smooth and accurate motion. The platform is experimentally validated in the laboratory through arm-based locomotion and coordinated base-arm movements using standard interfaces. The resulting control architecture can be used as the foundation for developing a common framework of the multi-arm robot in orbital application.

Keywords: Space Robotics · Control framework · Multi-Arm Robot · Assembly · Locomotion · ROS2 · IOSM.

1 Introduction

Multi-arm space robotics have gained prominence in In-orbit Servicing, Assembly and Manufacturing (ISAM) due to their dexterity and capability to perform complex tasks [1]. These robotic systems enable precise manipulation, making them suitable for applications such as satellite servicing, space station maintenance, and large-scale assembly [2]. The advantages of multi-arm robots over traditional single-arm systems include redundancy, improved payload handling, and the ability to execute coordinated motions [3].

The development of on-orbit servicing technologies has been extensively reviewed in prior studies, which highlight advancements in robotic autonomy, manipulation, and system-level architectures that support multi-arm robotic systems in space [4]. Research on in-orbit robotic control has highlighted the importance of optimal motion planning and task execution [5]. Closed-chain manipulation techniques have been explored to enhance stability in multi-arm coordination [6]. Moreover, real-time simulations in physics engines have provided insights into robotic behaviors in space and simulated space environments [2]. These facilitate the evaluation of control algorithms before real-world deployment, reducing mission risks and costs.

Inverse kinematics and optimization techniques have been utilized to enhance the performance of space robotic manipulators [7]. These are essential for tasks such as docking, component replacement, and structural assembly for ISAM. Furthermore, reconfigurable robotic arms have been proposed to enhance adaptability in dynamic space environments [8]. Thronson et al. discussed the transformative potential of robotic in-space servicing and how future systems could evolve toward higher levels of autonomy and modularity [9].

Despite these advancements, real-world validation of multi-arm locomotion remains a challenge. The in space simulation of the multi-arm robot's locomotion and tasks for ISAM are being explored in multiple studies [10] [11]. Current studies emphasize simulation without extensive hardware testing [3]. While multiple areas like cable-suspended robotic systems or robotic platforms with tracks are being explored to recreate space locomotions on the ground [12]. At Cranfield university, a Multi Arm Robot for In-orbit Operations (MARIO) has been developed to support the validation and verification of robotic and space systems in lab environment. This research aims to develop a ROS 2 control framework for a realistic representation of MARIO interactions using an easily extensible software architecture for research. This framework allows for efficient path planning and collision modelling for MARIO, ensuring that locomotion planning and execution are easier.

By leveraging ROS2 and moveit2, this study seeks to establish an efficient method for validating multi-arm locomotion strategies in a laboratory setup. The findings will contribute to future advancements in in-orbit assembly and servicing, improving the operational efficiency of space robotic systems.

Key Contribution: In the following sections we will first provide an overview of the critical issues of multi-arm robotic platforms for in-orbit assembly tasks, in particular to MARIO (section 2), moving towards the control architecture of the robotic system (section 3). Next, we describe the locomotion sequence adapted for the lab experimentation (section 4), then the results and observation are discussed in detail (section 5). Final remarks conclude the paper (section 6).

2 Problem Overview

Significant progress has been made in space robotics, but several key challenges remain. A primary difficulty is the accurate replication of zero-gravity conditions

in laboratory environments, which is essential for the effective execution of locomotion and space operations with robotic systems. Current methods typically involve suspending robots from above and using manipulators to simulate space operations [13].

However, these approaches restrict the robot’s motion and dynamic behavior. Achieving free-floating behavior, where the robot’s base and arms interact solely through internal forces without external constraints, cannot be accomplished with cables or tracks. As a result, it fails to fully capture the inertial effects and free body dynamics experienced in real space conditions, limiting the proper evaluation of the robot’s locomotion capability and overall system performances.

MARIO overcomes these limitations using an on-board pneumatic air-bearing platform, used in conjunction with a flat epoxy floor. This allows for the friction between the platform and the floor to be reduced to near zero, enabling a simulated planar microgravity environment. MARIO is then free to perform locomotion by docking to fixed structures in said environment. From a control perspective, in-situ planning of collision-free trajectories around these structures is non-trivial. The planner must consider collisions between MARIO’s body, three six-degree-of-freedom manipulators, and structures that MARIO moves relative to.

To solve this problem efficiently, a multi-layered control architecture is used to facilitate locomotion tasks: firstly, individual controllers are employed for each manipulator, enabling simple and solvable kinematic solutions. Secondly, a virtual environment representing the lab as collision objects is used to represent the planning space. This enables robust and efficient planning within an accurate representation of MARIO’s environment.

3 ROS Control Architecture Overview

The control system of MARIO is built on the Robot Operating System 2 (ROS2) framework, which provides a modular, real-time control system for coordinating the movements of the robotic platform. Each arm operates independently but is synchronized through ROS2 move group control nodes, ensuring smooth coordination during task execution. The control architecture is shown in Figure 1. It supports real-time communication between the sensors and actuators, providing continuous feedback for controlling the robot’s movement and task execution. The control architecture is divided into three macro-blocks:

- a *path-planner*, computes joint trajectories for the manipulators with self and environmental collision avoidance.
- the *ROS2 controllers*, the software-based controllers which calculate the required joint targets to execute and follow the target trajectories
- the *MARIO hardware interfaces*, a software layer that passes the joint targets calculated by the ROS2 controllers to the onboard hardware.

The experimental setup of MARIO in the Laboratory of Autonomous Navigation and Demonstration (LAND) is shown in Figure 2. Mario is positioned in

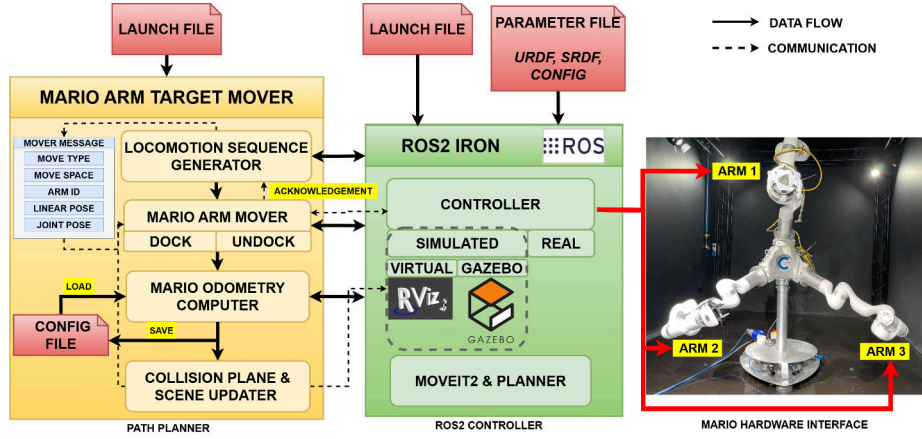


Fig. 1. MARIO Control Architecture

front of the spacecraft mock-up with multiple latching/docking points that are modelled into the ROS2 Framework for MARIO to do anchored arm-based locomotion. The SIROM interface from SENER [14] will take care of these anchoring interactions through visual servoing and ArUco markers. The locomotion is generated with the LAND walls and the spacecraft mock-up (except the docking points T1 to T10) as collision objects.

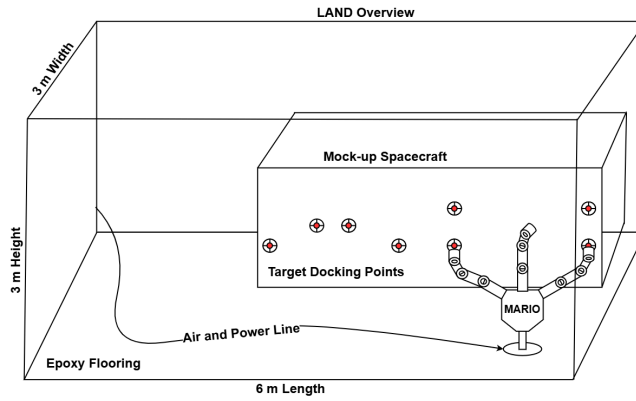


Fig. 2. MARIO Experimental Setup

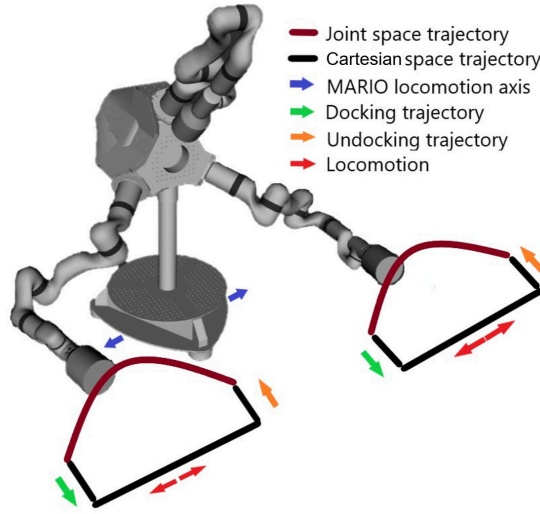


Fig. 3. MARIO control workflow

3.1 Locomotion Workflow for in-lab setup

The MARIO system employs a locomotion strategy that leverages its multi-arm configuration to "crawl" along surfaces. This method involves the robot using its arms to grab and release structural elements, pulling or pushing itself forward with the help of the SIROM interfaces. This approach allows MARIO to navigate complex space structures, such as the exterior of a satellite or space station, without relying on external propulsion or wheels. The same is replicated in the testing lab for MARIO to move along the surface of the spacecraft mock-up as shown in Figure 2.

MARIO's control system facilitates both docking and movement of the arms to achieve locomotion. When it needs to relocate, the system follows a precise sequence: firstly, the docked arm follows a trajectory that positions MARIO's body for the subsequent movements. The undocked arm then latches to a nearby interface allowing the other arm to release. This coordinated docking and undocking process allows MARIO to traverse the structure while maintaining stability during transitions. While one arm is docked, the control system determines the optimal path to the next docking interface for the undocked arm. It uses real-time feedback from the manipulator controllers to monitor the robot's position, ensuring accurate movement between docking points. As MARIO moves, the system continuously updates the collision environment to prevent obstacles. In order to ensure collision locomotion we need the position of collision objects used by the planner to be updated after each phase of MARIO's movement. This is achieved by with a localisation process that tracks MARIO's position using an odometry system.

3.2 MARIO Transformation Tree

MARIO's configuration consists of three manipulators connected to a central hub that often interact with shared objects or with each other. Each arm's motion impacts the others, creating a system of interdependent movements. This type of kinematic chain is complex to model and control, as it requires ensuring that all arms work in harmony without causing internal stresses or collisions. The frame description in Figure 4 shows the base frame that is tracked relative to the world frame. All trajectories and component meshes are modelled relative to this frame which serves as the origin for the entire MARIO assembly. The

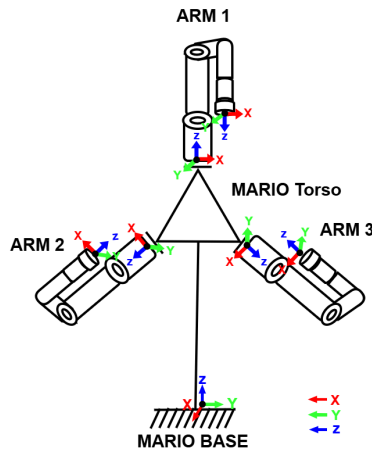


Fig. 4. MARIO Transformation Tree

locomotion sequence of the MARIO robot follows a structured workflow to ensure smooth and precise multi-arm movement, odometry computation, and collision avoidance. The basics behind the MARIO's crawling or inchworm movement along with the computational logic is explained in detail on the following sections.

4 MARIO Locomotion Sequence

The overall workflow which provides the path and target points for MARIO's locomotion and manipulation is illustrated in Figure 5, along with its message flow and subscriptions. The Moveit controller uses the OMPL planner with RRT algorithm for path planning. For the manipulator arm, the URDF provided with the XArm packages provided by Ufactory is being used for robotic controls.

4.1 Locomotion Sequence

The target points are modelled as target 1 to 10 in the controller and are used for locomotion. **Step 1: Compute Arm Movements and Update Dock**

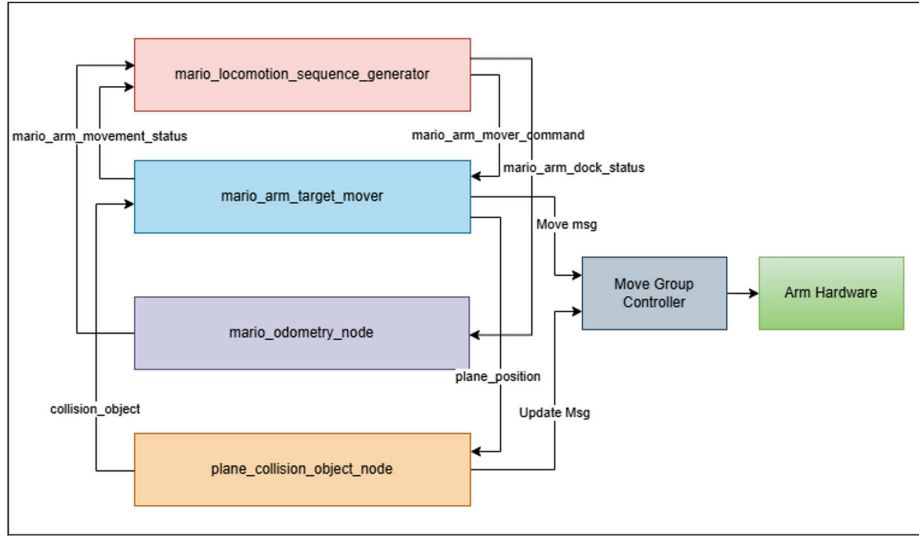


Fig. 5. MARIO Locomotion workflow

Status

The *mario_locomotion_sequence_generator* logs initial positions, calculates displacement, and updates transformation matrices: Mathematically, the transformation matrix T for the arm is represented as:

$$\begin{bmatrix} R(q) \\ p(q) \end{bmatrix} \quad (1)$$

where R is the rotation matrix and p is the translation vector. The base position update in the global coordinate frame is obtained as:

$$p_{new} = T \cdot p_{old} \quad (2)$$

where p_{old} is the previous position vector and p_{new} is the updated position.

Step 2: Verify Reachability and Execute Movement

The *mario_arm_target_mover* checks reachability in Cartesian or Joint space, updates collision planes, and determines movement using inverse kinematics: Mathematically, given a target position p_t , the required displacement in Cartesian space is given by:

$$\Delta p = p_t - p_{current} \quad (3)$$

If the movement is executed in joint space, inverse kinematics is used:

$$q = IK(p_t) \quad (4)$$

where q represents the joint configuration.

Step 3: Detect and Avoid Collisions

The *plane_collision_object_node* identifies obstacles from last updated collision objects positions via the Odometry data and refines trajectory planning by maintaining a minimum safe distance:

$$d_{min} \geq \epsilon \quad (5)$$

where d_{min} is the minimum distance from the arm to the obstacle and ϵ is a predefined safety threshold. Since LAND is a constrained lab environment, the minimum distance criterion is considered rather than the minimum energy for the trajectory calculations.

Step 4: Track Odometry and Apply Corrections

The *mario_odometry_node* records arm positions (Tool Center Point) before and after docking through joint values and FK, correcting the odometry model for the new update:

$$p_{odom} = T_{odom} \cdot p_{prev} \quad (6)$$

where T_{odom} represents the accumulated transformation matrix for odometry corrections.

Step 5: Execute Motion Commands The Move Group Controller transmits precise motion commands to the MoveIt2 controller, executing coordinated transformations:

$$T_{final} = T_{base} \cdot T_{arm} \cdot T_{end-effector} \quad (7)$$

where T_{base} represents the transformation from the world to the robot base, T_{arm} represents the transformation from the base to the arm joint, and $T_{end-effector}$ represents the final transformation to the end-effector.

This sequencer module and nodes will send the commands to MARIO controller to perform the planned trajectories of the locomotion sequence. The results are discussed in detail in the sections below.

5 Results and Discussion

The results of the MARIO locomotion sequence, as demonstrated using the proposed control architecture, will illustrate the robot's ability to perform controlled and repeatable movements. The virtual representation of the controller path planning and collision objects are given in the Rviz view as shown in 6. The locomotion sequence involved left and right movements of 350 mm, with collision objects dynamically updated to reflect MARIO's new positions.

The sequence begins with MARIO positioned at the center of the workspace, where Arm 2 docks and Arm 3 undocks, preparing for movement. Once Arm 2 is securely docked, only linear motion is executed to prevent potential damage to the setup. During the movement phase, Arm 2 moves to the right while MARIO's base shifts left. After returning to the initial position, the robot performs the reverse motion, where Arm 2 moves left and the base shifts right.

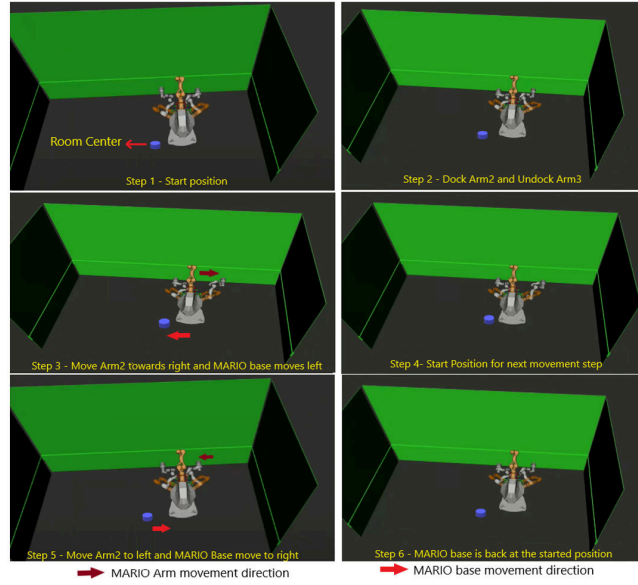


Fig. 6. MARIO locomotion simulation

This update mechanism ensures that the collision objects and information shifts around MARIO using odometry data, allowing the system to maintain an accurate position reference within the LAND environment.

The Figure 7 highlights key movements in the sequence with the hardware. The MARIO, once docked to one of the target points in the spacecraft mock-up, then receives a linear command to move Arm 2 750 mm along the negative y-axis in the tool frame, and the base is translated 750 mm along the positive Y-axis which is described in detail on the mario transformation tree shown in 4. It is done in 5 steps, with 150 mm or 250 mm movement commands being sent to the controller from the locomotion sequencer, while the collision information is being updated at the end of each set with the updated MARIO's base position. This controlled localization within the LAND environment enables the control system to execute and validate MARIO's locomotion effectively. Furthermore, this approach ensures that the MARIO system can perform real-time locomotion with its hardware counterpart.

The joint state analysis illustrated in Figure 8 will provide the overview of the anchored locomotion sequence representation in MARIO's Arm-2. The provided linear movements of the anchored locomotion will provide the smooth trajectory needed for the locomotion of MARIO since the joint space movement will be between the transition of initial points before each anchored motion and then the planner will slowly transition to the linear or Cartesian space coordinated movement, which is described in the plot.

The two key graphs, 9(a) and 9(b), illustrate MARIO's trajectory and odom-

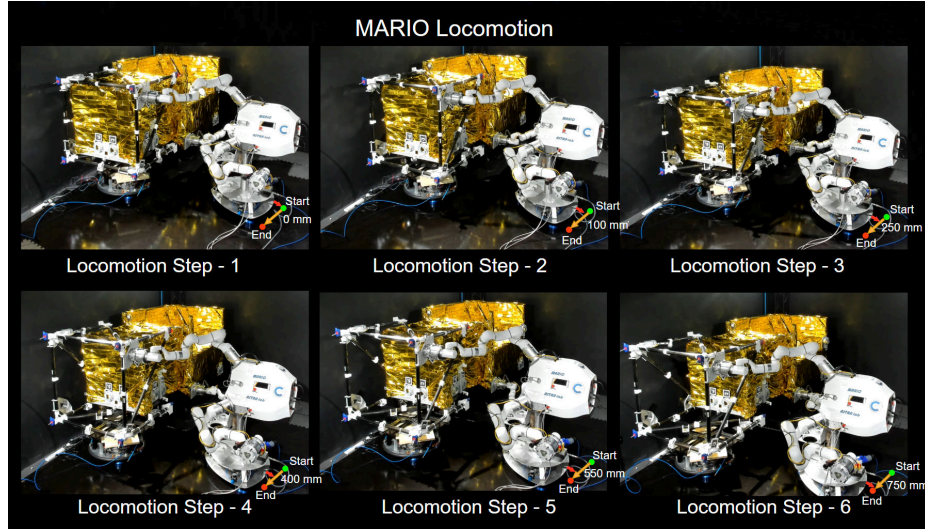


Fig. 7. MARIO Locomotion

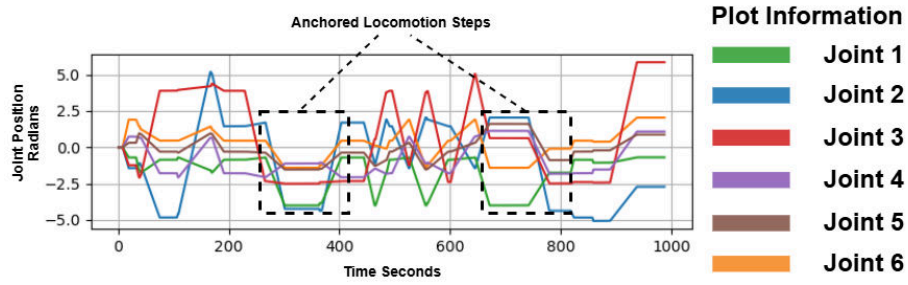


Fig. 8. Joint state plot of Arm-2 for Anchored Motion

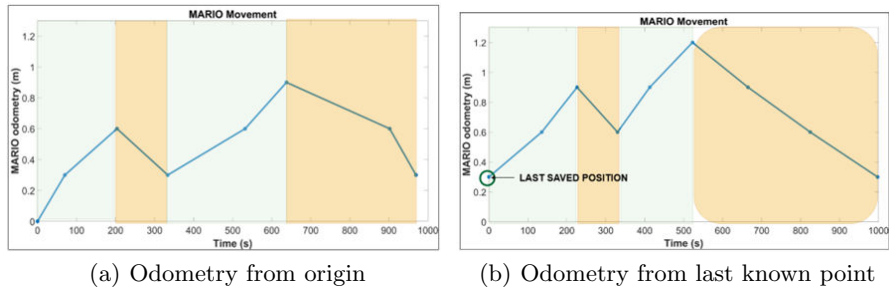


Fig. 9. Odometry Plot of MARIO locomotion

etry data within the LAND floor. These visualizations depict MARIO’s planar localization details, emphasizing displacement variations along the Y-axis. In the plot, pale green region represents leftward displacement (+Y direction), while the pale orange region denotes rightward displacement (-Y direction) of the MARIO base during the locomotion simulation. This distinction helps MARIO process relative objects and collision data, feeding essential path computation details into its controller. By leveraging odometry data, the LAND environment is computed as a collision object within the MoveIt control framework. This setup enables the environment to be translated within the plane while keeping the MARIO system fixed at the center, serving as both the world frame and origin. Additionally, this approach allows the system to verify planned paths and detect potential collisions through the in-situ planner before executing physical movements, ensuring safer operations. This alternative significantly enhances the robot’s awareness of the LAND environment and the positioning of interactive objects for path planning within the control framework, enabling more accurate validation of locomotion strategies.

6 Conclusion

The paper presented the control architecture for virtual simulation of the multi-arm robot to perform the locomotion and manipulation necessary for the ISAM application in a more adaptable framework for in-lab environment like the LAND. The collected odometry data during the tests demonstrated the effectiveness of the localization system in tracking MARIO’s position. The ability to update relative objects and collision information on MARIO’s arm movement after each step provides a dynamic and accurate representation of the operational lab(LAND) environment. This system allowed MARIO to maintain a known relative position to the structure and navigate efficiently within the simulated workspace. The results prove the feasibility of replicating frictionless conditions for MARIO and its environment for space robotics testing. This supports the development and validation of in-orbit servicing and manipulation tasks in laboratory environments. Future work will focus on introducing autonomy in MARIO’s locomotion and incorporating impedance control for docked interactions, further improving the platform’s capability to perform complex assembly and servicing operations with higher precision and safety.

References

1. K. McBryan, "Comparing Multi-Arm Robotics for In-Space Assembly," *Frontiers in Space Technologies*, vol. 3, Apr. 2022, doi: 10.3389/frspt.2022.702614.
2. Z. Xue, J. Liu, C. Wu, and Y. Tong, "Review of in-space assembly technologies," *Chinese Journal of Aeronautics*, vol. 34, no. 11, pp. 1–20, Nov. 2021, doi: 10.1016/j.cja.2020.09.043.

3. B. Ma, Z. Jiang, Y. Liu, and Z. Xie, "Advances in Space Robots for On-Orbit Servicing: A Comprehensive Review," *Advanced Intelligent Systems*, vol. 5, no. 1, Aug. 2023, doi: 10.1002/aisy.202200397.
4. A. Flores-Abad, O. Ma, K. Pham, and S. Ulrich, "A review of space robotics technologies for on-orbit servicing," *Progress in Aerospace Sciences*, vol. 68, pp. 1–26, 2014, doi: 10.1016/j.paerosci.2014.03.002.
5. J. P. Lutze, R. Schuller, H. Mishra, I. Rodríguez, and M. A. Roa, "Optimization of multi-arm robot locomotion to reduce satellite disturbances during in-orbit assembly," in *IEEE Aerospace Conference Proceedings*, IEEE Computer Society, 2023, doi: 10.1109/AERO55745.2023.10115776.
6. X. Chu, Q. Hu, and J. Zhang, "Path Planning and Collision Avoidance for a Multi-Arm Space Maneuverable Robot," *IEEE Trans Aerosp Electron Syst*, vol. 54, no. 1, pp. 217–232, Feb. 2018, doi: 10.1109/TAES.2017.2747938.
7. J. Zhao, X. Yang, Z. Zhao, G. Yang, and L. Zhao, "Inverse kinematics and multi-objective configuration optimization of the SSRMS manipulator," *Advances in Space Research*, vol. 72, no. 9, pp. 3580–3594, Nov. 2023, doi: 10.1016/j.asr.2023.06.058.
8. H. Mishra, M. De Stefano, and C. Ott, "Dynamics and Control of a Reconfigurable Multi-Arm Robot for In-Orbit Assembly," in *IFAC-PapersOnLine*, Elsevier B.V., Jul. 2022, pp. 235–240, doi: 10.1016/j.ifacol.2022.09.101.
9. H. Thronson, D. Akin, J. Grunsfeld, and D. Lester, "The Evolution and Promise of Robotic In-Space Servicing," in *AIAA SPACE 2009 Conference and Exposition*, Reston, Virginia: American Institute of Aeronautics and Astronautics, Sep. 2009, doi: 10.2514/6.2009-6545.
10. Bhadani S, Dillikar SR, Pradhan ON, et al., (2023) A ROS-based simulation and control framework for in-orbit multi-arm robot assembly operations. Presented at: ASTRA 2023: 17th Symposium on Advanced Space Technologies in Robotics and Automation, 18-20 October 2023, Leiden, The Netherlands, link:<https://dspace.lib.cranfield.ac.uk/handle/1826/20622>.
11. Dillikar SR, Leslie C, Upadhyay S, et al., (2025) A ROS-based control framework for simulating locomotion of a multi-arm space assembly robot. In: Towards Autonomous Robotic Systems, 25th Annual Conference, (TAROS 2024), 21-23 August 2024, London, UK, Part II. Lecture Notes in Computer Science, Volume 15052, pp. 93-105 doi:https://doi.org/10.1007/978-3-031-72062-8_9
12. Y. Gao and S. Chien, "Review on space robotics: Toward top-level science through space exploration," 2017. [Online]. Available: <https://www.science.org>
13. M. Deremetz et al., "MIRROR-A Modular and Relocatable Multi-arm Robot Demonstrator for On-orbit Large Telescope Assembly.", 17th Symposium on AASTRA At: Leiden link: <https://roboshare.esa.int/ASTRA/Astra2023/2023 PA-PERS/80.Deremetz.Design and Validation of a Modular Multi-arm Relocatable Robot for in-space Servicing and large Structure Assembly.pdf>
14. M. Díaz-Carrasco, A. Lázaro, A. Ruiz, M. Centeno, J. Gala, Javier Viñals, "SIROM Roadmap for Future In-Orbit Servicing Applications", in Proceedings of the ESA/ESTEC Workshop on Advanced Space Technologies for Robotics and Automation (ASTRA), Noordwijk, Netherlands, 2023



DSPACE

<https://dspace.org/>

Locomotion of Multi Arm Robot for In-Orbit Operations (MARIO) in lab environment

Elavazhagan, Praveen; Leslie, Cameron; Upadhyay, Saurabh; Tang, Gilbert; Felicetti, Leonard

2026-08-20

Springer

https://doi.org/10.1007/978-3-032-01486-3_18

Elavazhagan P, Leslie C, Upadhyay S, et al., (2026) Locomotion of Multi Arm Robot for In-Orbit Operations (MARIO) in Lab Environment. In: Towards Autonomous Robotic Systems, Lecture Notes in Computer Science, Springer Nature Switzerland, 2026, pp. 223-234. 26th Annual Conference, TAROS 2025, August 20-22 2025, York, UK