A ROS-BASED SIMULATION AND CONTROL FRAMEWORK FOR IN-ORBIT MULTI-ARM ROBOT ASSEMBLY OPERATIONS

Saksham Bhadani[1], Sairaj R Dillikar[2], Omkar N Pradhan[3], Irene Cotrina de los Mozos[4], Leonard Felicetti[5], Saurabh Upadhyay[6], and Gilbert Tang[7]

[1]*MSc Robotics, Cranfield University, Cranfield, United Kingdom, Email: saksham.bhadani.558@gmail.com*
[2]*MSc Robotics, Cranfield University, Cranfield, United Kingdom, Email: sairaj.dillikar.102@gmail.com*
[3]*MSc Robotics, Cranfield University, Cranfield, United Kingdom, Email: omkarnilesh.pradhan.867@gmail.com*
[4]*MSc Robotics, Cranfield University, Cranfield, United Kingdom, Email: i.cotrinadelosmozos.212@gmail.com*
[5]*School of Aerospace, Transport and Manufacturing, Cranfield University, Cranfield, United Kingdom, Email: Leonard.Felicetti@cranfield.ac.uk*
[6]*School of Aerospace, Transport and Manufacturing, Cranfield University, Cranfield, United Kingdom, Email: Saurabh.Upadhyay@cranfield.ac.uk*
[7]*School of Aerospace, Transport and Manufacturing, Cranfield University, Cranfield, United Kingdom, Email: g.tang@cranfield.ac.uk*

ABSTRACT

This paper develops a simulation and control framework for a multi-arm robot performing in-orbit assembly. The framework considers the robot locomotion on the assembled structure, the assembly planning, and multi-arm control. An inchworm motion is mimicked using a sequential docking approach to achieve locomotion. An RRT* based approach is implemented to complete the sequential assembly as well as the locomotion of MARIO across the structure. A semi-centralised controller model is used to control the robotic arms for these operations. The architecture uses MoveIt! libraries, Gazebo simulator and Python to simulate the desired locomotion and assembly tasks. The simulation results validate the viability of the developed framework.

Key words: Robot Operating System (ROS); On-orbit Assembly; Locomotion; Free-Floating nature; Multi-Manipulator System; Multi-Armed Robot for In-Orbit Operations; Spatial Reticular Structure (SRS); Modular Truss Structure; Path planning.

1. INTRODUCTION

Large scale space infrastructures such as space stations and deep-space telescopes have played a vital role in building a fundamental understanding of our universe, and have also been key in the development of space-capable technologies. The size of these structures is currently limited by the fairing capacity and constraints of launch vehicles. With the present state of technology, assembly in space is the most viable way around these limitations. Moreover, future space constructions will require in-orbit robotic assembly operations, especially in ambitious projects that require the building and maintenance of very large structures such as space-based solar power plants or cosmic telescopes. Multi-Manipulator Systems (MMS) are preferred for space robotics due to their improved object handling and performance compared to single-arm robots. For example, [1] proposed the Multi-arm Installation Robot for readying Orbital Replacement Units (ORU) and Reflectors, or MIRROR for short. It comprises a torso and three identical arms with Standard Interfaces (SI) as end-effectors for object manipulation and remarkably for the robot's locomotion. The RAMST (Robotically Assembled Modular Space Telescope) robot, for its part, attempts to include six arms. Hence, it is called hexbot [2]. This increase in degrees of freedom allows for the minimisation of sudden changes in the displacement of the centre of mass and the reduction of transmitted loads into the structure.

Independently of the system morphology, SIs are key to the interaction of the robot with its environment. Different proposals to achieve primarily mechanical, data and power connections can be found throughout the years. In [3], the evolution of the docking mechanisms used in spacecraft docking is explained. The Apollo vehicles achieved docking using a probe and a cone-latching system. This system is light in weight and hence is still used. However, they require high contact velocity for docking. NASA's docking, a peripheral capture system, is much more versatile due to the system's flexibility and having six telescope-based actuation arms that facilitate the shifting of the mating adapter. This comes with the cost of additional weight but allows the soft capturing of the system. Recent developments set the focus on creating a standard uniform docking interface for European missions, as seen in [4], [5] [6].

When using multiple manipulators together, various approaches have been researched extensively. For this particular use case, control concepts from swarm robotics were considered for adaptation. The primary approaches being centralised control, (where all manipulators use a single controller/driver), semi-centralised control (where all manipulators have their independent controllers but at the same time have another singular controller commanding them and acting as the main one), and de-centralised control (where each manipulator has its own independent controller). Since multiple independent arms are present, decentralised control can be practical for space robots [7], [8]. It allows the units to have separate power while communicating with the control systems. On the other hand, a centralised strategy combines the control of all the manipulators onto a single computer, which can be computationally expensive depending on the use case. Nevertheless, it is still a feasible approach as it can ensure all agents (robotic arms) are in sync [9]. Semi-centralised control can also be adequate if a method similar to the one followed in [10] is used, where the moving cost for the tasks is minimised and only necessary movements are made.

Depending on the mechanism and configuration used, the control equation can result in different singularity conditions, which often makes the control requirements unique. The kinematics and dynamics of a robot with multiple manipulators can be challenging since the calculation changes depending on the end-effectors' docking. If one manipulator is docked, its end-effector acts as the base frame for the entire system, whereas when two manipulators are docked onto a fixed structure, they form a closed-loop kinematic chain, which has entirely different kinematic and dynamic properties. In the closed-loop kinematic chain, if a torque is applied at one of the joints, it is transferred to all the others since all the links are rigid. In order to avoid damaging the joint motors, all of them must be actuated to nullify the force [11]. In dynamic movement, where the motion between two manipulators is coordinated, approaches such as trajectory coordination using kinematic constraint equations based on the kinematic constraints between end-effectors can be used, as explained in [12].

The ease of solving kinematic equations can change depending on the arm configurations and link lengths, leading to the MMS design. The latter presents significant challenges in space conditions due to the harsh and uncertain environment, the long communication delays, and the limited availability of resources. The design should consider the specific mission requirements, such as the type of tasks to be performed, the mission's duration, and the available resources. When it comes to implementing and testing autonomous robots under the extreme conditions of outer space, a lot of critical capabilities, mission planning, etc. need to be designed and validated. Therefore, simulation can play a vital role in the starting phases. In addition, these simulation tools can flag problems in the early stages of projects hence they can save considerable research and development time and costs.

Most cutting-edge research on robotics uses ROS as a basis due to its high reliability, compatibility, and open-source libraries. One of the prominent use cases of ROS in space robotics can be observed in Robonaut 2 [13], developed by NASA and General Motors. Other NASA missions, such as Astrobee [13], used ROS too. Gazebo is another powerful open-source tool: it supports the simulation of mobile robots, as presented in [14], manipulator tasks as discussed in [15], and even aerial robotic simulations, as shown in [16]. Efforts have been made to adopt ROS and Gazebo framework into a unified library for space robotics as discussed in [17], but further development on control is needed for a better adaptation towards creating a framework capable of supporting multi-manipulator systems for complex tasks.

This paper presents the results of an investigation performed by a team of students at Cranfield University on the adaptability of the Robotic Operating System (ROS) and Gazebo environment to perform in-orbit assembly simulations. Taking reference from under-development concepts, the simulation scenario considers the challenge of the autonomous assembly of a modular truss structure using a Multi-Arm Robot for In-orbit Operations (MARIO) in free-floating conditions, the main operations being locomotion and assembly. The control interface made use of open-source kinematic solution algorithms and trajectory planning incorporated through MoveIt! libraries.

2. SCENARIO DEFINITION

This section aims to explain the presented work developed at fundamental and conceptual levels. The system design starts with the definition of parameters and constraints around the dimensions, functional and operational requirements and control requirements. Considering the set of conditions based on the capability to locomote and assemble, a planar delta configuration i.e. all of the manipulator's base frames being in the same plane in a 120-degree equidistant placement was chosen. This configuration allows for good independence as well as a collaborative work envelope for manipulators, which is essential for the desired use case.

2.1. MARIO's Configuration

The configuration of the robotic arm was derived based on an industrial collaborative articulated robotic arm Kinova Gen 3 which is equipped with six revolute joints connected serially. The frame assignments for the individual joints were realised according to the classical DH convention. Figure 1 represents the frames of the final assembly of the MARIO. Figure 2a gives a graphical representation visualising all three robotic arms' frames along with the SI.

The DH parameter, as illustrated in Table 1, was tabulated for the first robot arm, and for the subsequent arms, only

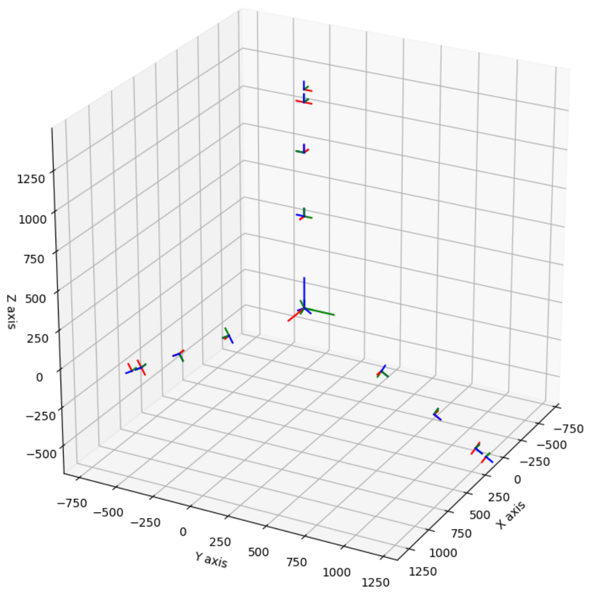


Figure 1: Representation of Joint Frames for MARIO with three Kinova Gen 3 manipulators

the base frame is required to rotate by $\pm 120°$ as shown in Table 2 and Table 3, and the frames for the rest of the joints are same.

Table 1: DH Parameter table for Kinova Gen 3 manipulator (Arm-1 of MARIO)

ith ***Joint***	α_{i-1}	a_{i-1}	d_i	θ_i
0a	0	0	0	0
1	0	0	604.16mm	θ_1
2	90°	0	0	θ_2
3	0	410mm	0	$\theta_3 + 90°$
4	90°	0	0	$\theta_4 + 90°$
5	90°	0	320mm	$\theta_5 + 90°$
6	90°	0	0	$\theta_6 + 180°$
7	0	0	80mm	0

Table 2: DH Parameter table for Base Joint (Arm-2 of MARIO)

i^{th} ***Joint***	α_{i-1}	a_{i-1}	d_i	θ_i
0b	120°	0	0	0

Hexagonal Spatial Reticular Structures (SRSs) are the modules used to make up said truss structure that can be seen in 2b, allowing for high strength and ability of tessellation. A custom-designed SI is used on the robot's end-effectors and the SRSs to enable docking. MARIO uses these SIs for assembling and locomoting on the structure. The corresponding models can be seen in Figure 2.

In a typical operation, two robotic arms are used for locomotion following an inchworm gait strategy, while the third arm is used to carry the SRSs to assemble the space structure. The overall scenario also considers a primary hub, which hosts the SRS modules that MARIO can collect, transport and place to build the structure, as shown in Figure 2c.

Table 3: DH Parameter table for Base Joint (Arm-3 of MARIO)

i^{th} ***Joint***	α_{i-1}	a_{i-1}	d_i	θ_i
0c	$-120°$	0	0	0

2.2. Concept of Operations (CONOPS)

The mission takes place as follows. The hub, containing MARIO and the SRSs, is launched into space. Once in orbit, MARIO emerges and starts its regular operation. This begins by fetching the first SRS from the then-open top of the hub with one of the arms. Using the remaining two robotic arms, MARIO climbs down the front wall by means of the docking points placed for this purpose. Now, at the bottom, the robot assembles the module at the first assembly position: the central or "zeroth" module located right below the storage area.

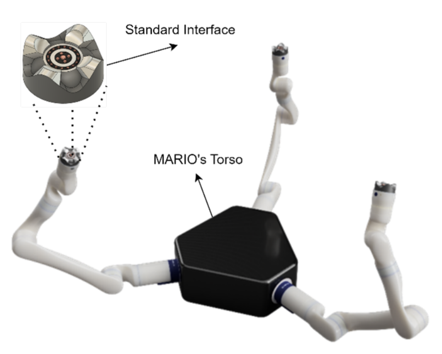
The entire assembly takes place in rings around the central hub. Afterwards, MARIO disengages and climbs back up to pick up the next SRS module. This cycle, represented in Figure 3, is repeated until the structure's completion. The difference between iterations is the assembly point, which depends on the SRS currently being placed and is given by a coverage planning algorithm discussed in Section 2.4. To get to that desired location, MARIO crawls on the previously assembled modules using its two manipulators: their end-effector SIs are docked with the upper SIs of the assembled SRSs. The specific sequence would be obtained by the path planning algorithm.

2.3. ROS-Gazebo simulation & control architecture

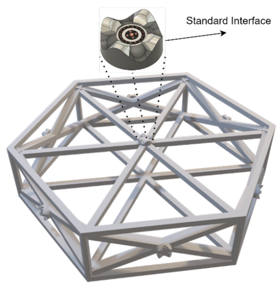
The simulation has been developed on ROS1-Noetic, and Gazebo used to visualise them. The robot configuration packages are generated through MoveIt! Setup Assistant using the MARIO URDF[1] packages. Python scripts comprising functions responsible for the sequence of tasks to be completed are used for performing the mission. A Gazebo plugin named ROS-Link-Attacher [18] is utilised to simulate the docking of the robot arms with the SRSs during the locomotion and assembly tasks. A tree search algorithm, based on the RRT*, is proposed to generate an optimised path through the already-built structure to locate and assemble new SRSs.

The control strategy of MARIO is inspired by decentralised approaches in swarm robotics, wherein its three arms are treated as individual robots. For the presented Gazebo simulation results, kinematic solvers available in MoveIt! have been used, along with predefined poses to follow a set of known functioning robot configurations.

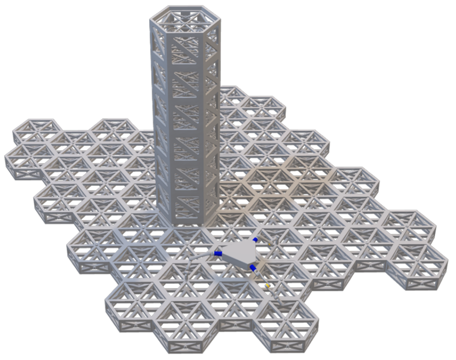
[1]URDF - Unified Robotic Description Format



(a) Multi-Arm Robot for In-Orbit Operations (MARIO)



(b) Spatial Reticular Structure (SRS)



(c) Central Hub Structure with the SRSs being assembled

Figure 2: Graphical representations of MARIO, SRS, and Central Hub

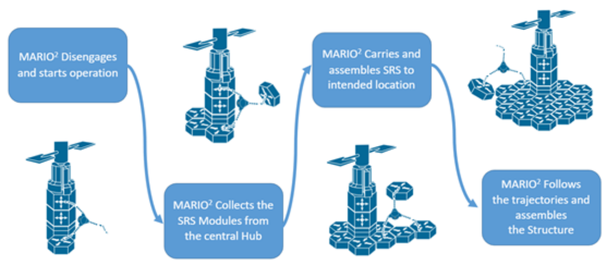

MARIO^2 Disengages and starts operation
MARIO^2 Collects the SRS Modules from the central Hub
MARIO^2 Carries and assembles SRS to intended location
MARIO^2 Follows the trajectories and assembles the Structure

Figure 3: Concept of Operations

But, for achieving full autonomy, the approach with separate solvers for each manipulator would be more suited.

This approach obtains the accurate location of the torso using forward kinematics in combination with the inversion of the kinematic chain for a single manipulator which is 'docked' to the space structure. The second arm's joint values can then be calculated through inverse kinematics based on the location of the torso. This allows for a relatively simpler control since only a few viable solutions are generated through calculations. In the case of two arms being treated as a single kinematic chain, the combined DOF (degrees of freedom) of the two manipulators results in a kinematic chain of 12 DOF. When kinematics for this configuration is solved, many solutions are generated, creating an additional requirement for a cost-function-based optimization algorithm. The robustness of this strategy has been tested in simulations where the main structure is in free-floating conditions.

2.3.1. Control Architecture for ROS-based Simulation

The URDF packages were generated from the detailed design of the systems. The initial step was to set up the configuration files required to run the simulation. These packages were MARIO (the 3-arm robot system), and the SRS module (spawning of individual SRS). The relevant ROS parameters are loaded into the ROS server from these URDF packages and the YAML (robot controller) as shown in Figure 4.

The joint state controller and joint trajectory controller were utilised within the controller configuration files (YAML) of the URDF package. The *joint_state_controller* is responsible for reading the joint state of all the existing joints and is of topic type: *sensor_msgs/JointState* [19]. Additionally, one of the essential controllers is *JointTrajectoryController* under *position_controller*, which receives a position input and generates an entire trajectory. On the other hand, within MoveIt! ROS controllers are also defined as individual arm groups, and the controller type is used as *FollowJointTrajectory*.

A MoveIt! library was used to generate the movegroup configuration packages that contain the motion planning and other framework files. One of the most significant parameters was gravity, which was kept at zero for the x, y, & z directions with respect to the world frame. The simulation was performed using Python scripts, and it is comprised of two parts: locomotion and assembly of a single SRS module. Multiple functions are defined within the scripts for attaching, crawling and spawning. The coupling of two bodies requires the plugin of type *ros_link_attacher.so*, which was utilised from the ros link attacher package. This plugin is introduced within a world file, as the Gazebo would load all the necessary plugins required for the simulation.

2.3.2. Autonomous Control Architecture

During normal use of a manipulator, with the base frame location known and the pose (position and orientation) of the target known, inverse kinematics can be used for calculating the required joint values as shown in [20]. These joint values can then be supplied to the joints to achieve the desired manipulator configuration.

For solving forward kinematics, the joint angles are required, and the result is the position of the TCP (Tool Centre Point) of the manipulator. Using the same DH pa-

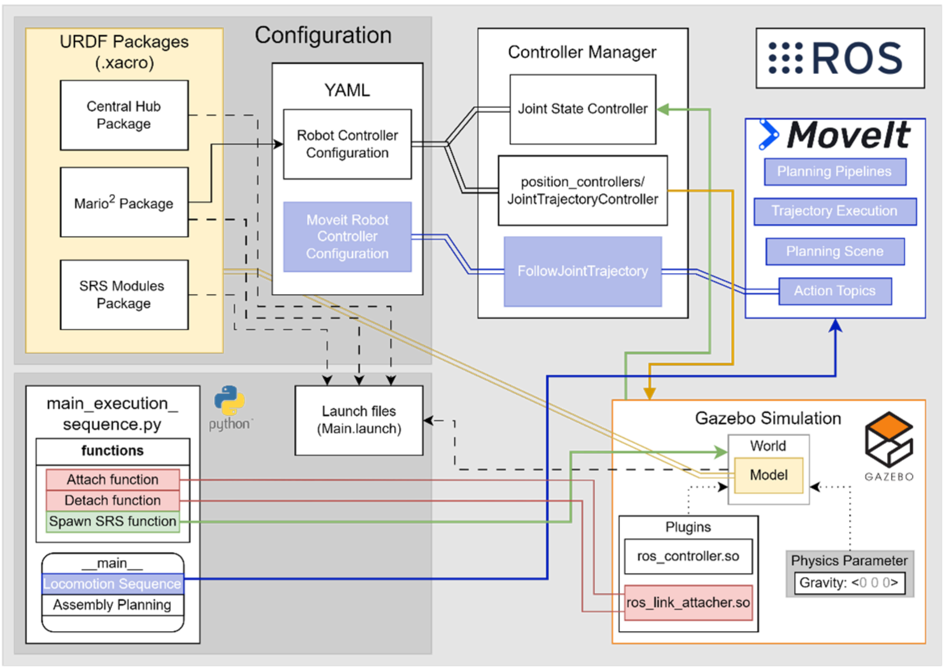


Figure 4: ROS Simulation Architecture

rameters mentioned in Table 1, 2, 3, the exact location of the manipulator final frame can be calculated if the position of the other extremity is known, and the joint angles can be acquired.

ROS, as we know, utilises nodes, topics and services connected to each other as the skeletal functional architecture. In case of the simulation, ROS controllers are being used for controlling individual joints. Using scripts and proprietary commands, the values of the joint angles can be modified/changed, which in turn is reflected in Gazebo. From Gazebo simulation, the instantaneous state of the joints can be determined using the joint state service.

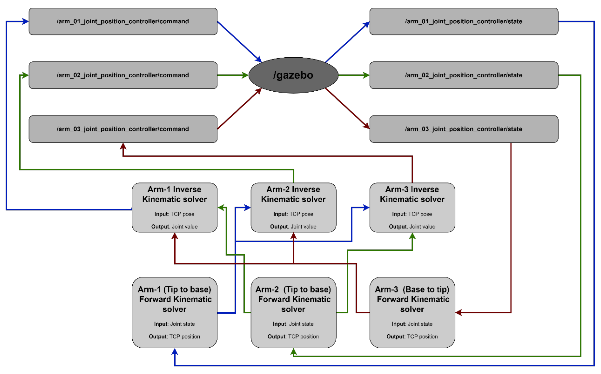

/arm_01_joint_position_controller/command
/arm_02_joint_position_controller/command
/arm_03_joint_position_controller/command
/gazebo
/arm_01_joint_position_controller/state
/arm_02_joint_position_controller/state
/arm_03_joint_position_controller/state
Arm-1 Inverse Kinematic solver
Input: TCP pose
Output: Joint value
Arm-2 Inverse Kinematic solver
Input: TCP pose
Output: Joint value
Arm-3 Inverse Kinematic solver
Input: TCP pose
Output: Joint value
Arm-1 (Tip to base) Forward Kinematic solver
Input: Joint state
Output: TCP position
Arm-2 (Tip to base) Forward Kinematic solver
Input: Joint state
Output: TCP position
Arm-3 (Base to tip) Forward Kinematic solver
Input: Joint state
Output: TCP position

Figure 5: Architecture for Automation of MARIO

Combining the inverse and forward kinematics as in Figure 5, MARIO can calculate its position and control the other manipulators based on it. An essential requirement to implement this method of operation is the inversion of frames for the manipulators. As explained in Figure 6, the base frame and end-effector frames would be swapped when the locomotion is done for either end. Considering the movement of one of the arms while the other is docked, the docked manipulator's end effector frame is treated as a base frame, the position of the torso of the robot is calculated using forward kinematics, and then the inverse kinematics for manipulation of the other arm can be calculated considering the torso as the base frame.

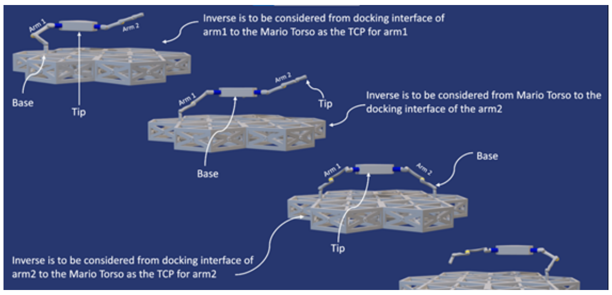

Arm 1
Arm 2
Inverse is to be considered from docking interface of arm1 to the Mario Torso as the TCP for arm1
Base
Tip
Inverse is to be considered from Mario Torso to the docking interface of the arm2
Base
Tip
Base
Tip
Inverse is to be considered from docking interface of arm2 to the Mario Torso as the TCP for arm2

Figure 6: Concept of inversion of frame assignment

2.4. Path Planning and locomotion

To effectively navigate the MARIO to the destined point of assembly, an RRT* based algorithm was developed and tested on MATLAB to obtain the waypoints for each SRS module with respect to the central hub. The algorithm starts by generating paths based on all the available SI locations on the structure as well as its dimensions. Once this is done, all the generated paths are taken into consideration, and based on the 'weight' of each element, the ’cost’ for each route is derived. Considering the fact that the ’cost function’ is nothing but the assigned weight multiplied by the number of elements, the path with the least cost is chosen as it represents the most efficient viable path. When the complete structure is assembled, the location of all the assembly points can be obtained relative to the global origin. Considering the number of concentric rings that need to be assembled, all the assembly points are collected and then sorted according to the order that completes individual rings at a time with the inward-out coverage.

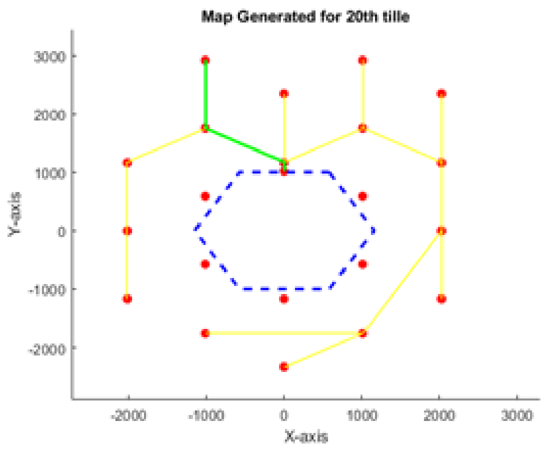


Figure 7: Path generated for 20th SRS using RRT* based algorithm computed on MATLAB

The problem that needs to be considered is that the points available for MARIO to traverse depend on the number of SRS modules already assembled in the previous steps, as they provide the physical points for intermediate docking to locomote on. The algorithm also accounts for the same and creates a new map for every iteration of available points from the already assembled SRS modules and thus provides the next point of assembly.

In addition, the shortest path from the 0th(zeroth) docking module to the assembly point is maintained, decreasing the number of steps to carry out for the assembly and thus reducing the overall cost. An example of a path generated for 20th SRS is shown in Figure 7. The Numerical results from RRT* have been shown in Table 4 consisting of values generated to traverse to and from the desired SRS. These values are represented in a cartesian coordinate system for both *Arm-1* and *Arm-2* demonstrating a sensible navigation path.

3. SIMULATION AND RESULTS

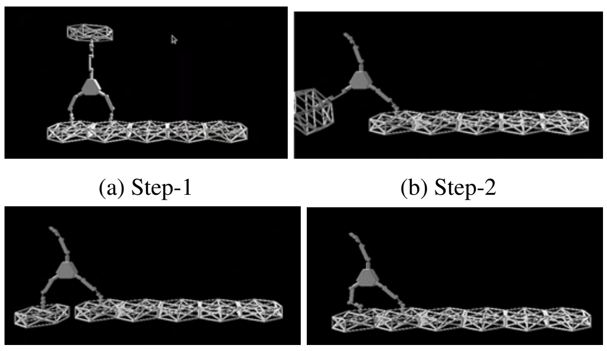
The locomotion cycle consists of predefined robot poses for two arms (mainly to perform the inchworm mechanism); these poses are defined as lift pose and stand pose, so one single locomotion cycle would incorporate a sequence of *Arm-1* and *Arm-2* with their respective poses, along with the docking function. The predefined poses

Table 4: Sequence of steps generated for MARIO’s Arm-1 & Arm-2 up until the 20th SRS in Cartesian Coordinate System

To Traverse Forward					
	Arm-1			*Arm-2*	
X	Y	Z	X	Y	Z
0.45	6321.19	1021.1	0.45	5120.19	1021.1
0.45	5120.19	1021.1	0.45	3920.19	1021.1
0.45	3920.19	1021.1	0.45	2720.19	1021.1
0.45	2720.19	1021.1	0.45	1520.19	1021.1
0.45	1520.19	1021.1	0	316.1	1170
0	316.1	1170	-1013.25	316.1	1755
-1013.25	316.1	1755	-1013.31	316.1	2925
To Traverse Backwards					
	Arm-1			*Arm-2*	
X	Y	Z	X	Y	Z
0	316.1	1170	-1013.25	316.1	1755
0.45	1520.19	1021.1	0	316.1	1170
0.45	2720.19	1021.1	0.45	1520.19	1021.1
0.45	3920.19	1021.1	0.45	2720.19	1021.1
0.45	5120.19	1021.1	0.45	3920.19	1021.1
0.45	6321.19	1021.1	0.45	5120.19	1021.1

are then called with the help of the move group commander service and are executed by the plan group method.

The assembly planning method is similar to the locomotion cycle, but the robot poses are defined for the arms that involve the docking of the spawned SRS module (or Nth module) with the previously assembled one. At the end of each assembling cycle, an SRS is spawned at a specified location in the SDF[2] file of the SRS modules’ URDF package. The spawning is usually triggered by calling the *SpawnModel* service type associated with the *spawn_sdf_model* service.



(a) Step-1 (b) Step-2

(c) Step-3 (d) Step-4

Figure 8: ith SRS Assembly Steps

To further analyse the sequencing of the robotic arms and the movement of MARIO the output data and various frames were observed. The relative position of the base frame of MARIO with respect to the world frame was plotted. Figure 9 shows that the robotic system is able to locomote using the control architecture hence validating the working of the same.

Furthermore, the position of the end-effector (the SI attached to the manipulators) during locomotion, as well as the joint values for each respective joint were plot-

[2]SDF - Simulation Description Format

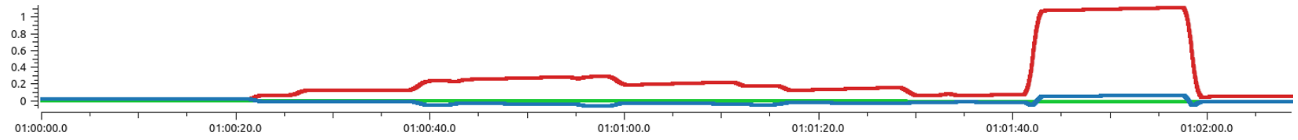
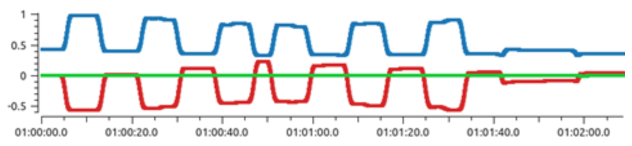
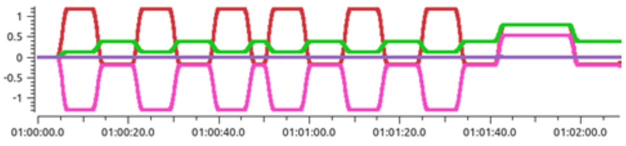
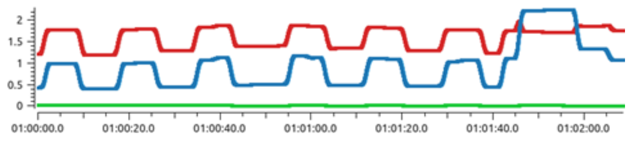


Figure 9: Base frame of the MARIO w.r.t world frame (Distance as a function of Simulation Time)


(a) Arm-1 End-Effector Position w.r.t MARIO's Base Frame


(b) Arm-1 Joint Values


(c) Arm-2 End-Effector Position w.r.t MARIO's Base Frame

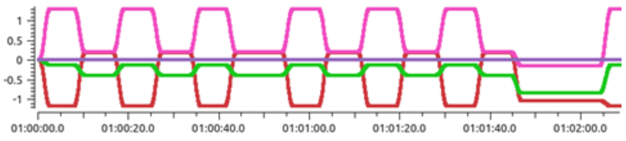

(d) Arm-2 Joint Values

Figure 10: Position and Joint Values of Arm-1 and Arm-2 (Distance/Joint angles as a function of Simulation Time)

ted using the PlotJuggler[3]. The simulation results can be conceived through the obtained plots as depicted in Figure 10. In addition to this, the simulation video[4] demonstrates the operations of locomotion and assembly.

3.1. Technical Specifications

The hardware utilised was a computer system with an Intel(R) Core^{TM} i7-9750H CPU @ 2.60GHz processor paired with 16 GB RAM and an NVIDIA GeForce GTX 1050 graphics card on Ubuntu 2020.04 platform. Also, the project was developed on Python 3.7 and the simulation was validated on ROS Noetic Ninjemys 8 with Gazebo v11.11.

4. CONCLUSIONS AND FUTURE WORK

The open-source nature of ROS and the community support makes it relatively easy to develop and integrate complex robotic systems onto it. This paper presented a viable ROS-based framework, which was developed for controlling the proposed multi-manipulator system. The prerequisites and parallel development have also been explained for the entire architecture. A fully autonomous operation of the system is explored further through a separate strategy demonstrating the ease of adaptation of the approach to fit the requirements. For the path planning, the outputs from an RRT* based algorithm have also been discussed. Finally, using all the aforementioned elements, the simulation results from Gazebo are presented proving the feasibility of the framework. The work carried out for the project can be accessed through the Space Robotics Lab Cranfield University repository at [21].

For a more realistic simulation, the gravity gradient acting upon the orbiting bodies would have to be taken into account. This can be integrated through the On-Orbit-ROS package [22]. On another note, Visual Servoing (VS) could be added to enhance the approach manoeuvre by allowing the robotic arm to locate the SI with which to couple. Moreover, a more sophisticated plug-in that demonstrates the actual mechanical latching could be developed. Finally, with a view to reduce the assembly time, the work could be carried out by multiple MARIOs with coordination among them and simulated by defining each MARIO under different namespaces in the ROS architecture.

Cranfield Space Robotics Laboratory is carrying out further development for MARIO and other space robotic systems, both on control software and physical hardware, with the aim to establish a potential future research in the Space Robotics field.

[3]PlotJuggler - https://plotjuggler.io/

[4]Simulation Video - https://youtu.be/AsthlnugMb4

REFERENCES

1. J Estremera, J García, P Romeo, A Rodriguez, I Colmenarejo, M Lucia, A Rusconi, G Sangiovanni, S Cordasco, D Antonucci, A Margan, L Baccielliere, N G Tsagarakis, J Viñals, G Guerra, and L Gerdes. Re-locatable manipulator for on-orbit assembly and servicing.

2. Nicolas N Lee, Joel W Burdick, Paul Backes, Sergio Pellegrino, Kristina Hogstrom, Christine Fuller, Brett Kennedy, Junggon Kim, Rudranarayan Mukherjee, Carl Seubert, and Yen-Hung Wu. Architecture for in-space robotic assembly of a modular space tele-

scope. *https://doi.org/10.1117/1.JATIS.2.4.041207*, 2:041207, 07 2016.

3. Heather Hinkel, John J Zipay, Matthew Strube, and Scott Cryan. Technology development of automated rendezvous and docking/capture sensors and docking mechanism for the asteroid redirect crewed mission. *IEEE Aerospace Conference Proceedings*, 2016-June, 06 2016.

4. Martin Kortmann, Kai-Uwe Schröder, Andreas Dueck, Christopher Zeis, Tobias Meinert, and Kai-Uwe Schroeder. Design and qualification of a multifunctional interface for modular satellite systems homer-highly redundant modular robotic systems for space applications view project design and qualification of a multifunctional interface for modular satellite systems. *https://www.researchgate.net/publication/328314664*, pages 1–5, 2018.

5. Stefan Scherzinger, Jakob Weinland, Robert Wilbrandt, Pascal Becker, Arne Roennau, and Rüdiger Dillmann. A walking space robot for on-orbit satellite servicing: The recobot. *http://arxiv.org/abs/2203.10217*, 03 2022.

6. Juan Sánchez, García Casarrubios, Pierre Letier, Torsten Siedel, Mathieu Deremetz, Edgars Pavlovskis, Benoit Lietaer, Korbinian Nottensteiner, Maximo A Roa, Juan Sánchez, Javier Luis, Corella Romero, and Jeremi Gancet. Hotdock: Design and validation of a new generation of standard robotic interface for on-orbit servicing. *https://www.researchgate.net/publication/344871962*.

7. Yassine Bouteraa, Jawhar Ghommam, Gérard Poisson, and Nabil Derbel. Distributed synchronization control to trajectory tracking of multiple robot manipulators. *Journal of Robotics*, 2011:1–10, 2011.

8. A Prévot, R Gourdeau, F Aghili, and J C Piedboeuf. Multi-manipulator system cooperation in the perspective of dextre. 2004.

9. Behrokh Khoshnevis and George Bekey. Centralized sensing and control of multiple mobile robots. *Computers & Industrial Engineering*, 35:503–506, 12 1998.

10. Shuo Wan, Jiaxun Lu, and Pingyi Fan. Semi-centralized control for multi robot formation. *2017 2nd International Conference on Robotics and Automation Engineering, ICRAE 2017*, 2017-December:31–36, 02 2018.

11. Zeyuan Sun, Hong Yang, Que Dong, Yang Mo, Hui Li, and Zhihong Jiang. Autonomous assembly method of 3-arm robot to fix the multipin and hole load plate on a space station. *Space: Science & Technology*, 2021, 01 2021.

12. Chunjian Su, Shuai Zhang, Shumei Lou, Rui Wang, Gaohua Cao, Longyun Yang, and Qing Wang. Trajectory coordination for a cooperative multi-manipulator system and dynamic simulation error analysis. *Robotics and Autonomous Systems*, 131:103588, 09 2020.

13. M A Diftler, J S Mehling, M E Abdallah, N A Radford, L B Bridgwater, A M Sanders, R S Askew, D M Linn, J D Yamokoski, F A Permenter, B K Hargrave, R Platt, R T Savely, and R O Ambrose. Robonaut 2 - the first humanoid robot in space. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2178–2183, 2011.

14. Kenta Takaya, Toshinori Asai, Valeri Kroumov, and Florentin Smarandache. Simulation environment for mobile robots testing using ros and gazebo. *2016 20th International Conference on System Theory, Control and Computing, ICSTCC 2016 - Joint Conference of SINTES 20, SACCS 16, SIMSIS 20 - Proceedings*, pages 96–101, 12 2016.

15. Wei Qian, Zeyang Xia, Jing Xiong, Yangzhou Gan, Yangchao Guo, Shaokui Weng, Hao Deng, Ying Hu, and Jianwei Zhang. Manipulation task simulation using ros and gazebo. *2014 IEEE International Conference on Robotics and Biomimetics, IEEE ROBIO 2014*, pages 2594–2598, 04 2014.

16. Carlos Sampedro, Alejandro Rodriguez-Ramos, Bavle Hriday, Adrian Carrio, Paloma, and Pascual Campoy. A fully-autonomous aerial robot for search and rescue applications in indoor environments using learning-based techniques. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 95:601–627, 08 2019.

17. J L Ramón, J Pomares, and L Felicetti. A ros/gazebo-based framework for simulation and control of on-orbit robotic systems. 2022.

18. PAL Robotics. Gazebo ros link attacher. *https://github.com/pal-robotics/gazebo_ros_link_attacher*.

19. Sachin Chitta, Eitan Marder-Eppstein, Wim Meeussen, Vijay Pradeep, Adolfo Rodríguez Tsouroukdissian, Jonathan Bohren, David Coleman, Bence Magyar, Gennaro Raiola, Mathias Lüdtke, and Enrique Fernandez Perdomo. ros_control: A generic and simple control framework for ros. *The Journal of Open Source Software*, 2:456, 12 2017.

20. John J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Longman Publishing Co., Inc., USA, 2nd edition, 1989.

21. Saksham Bhadani, Sairaj R Dillikar, Omkar N Pradhan, Irene Cotrina de los Mozos, Leonard Felicetti, Saurabh Upadhyay, and Gilbert Tang. Space-robotics-lab-cranfield-university/gdp-rbt-2023-group-6. *https://github.com/Space-Robotics-Lab-Cranfield-University/GDP-RBT-2023-Group-6*.

22. J L Ramón, J Pomares, and L Felicetti. Onorbitros/simulation: Ros framework for on orbiting space robots simulations. July 2022. Accessed on 11/05/2023.