Docking Control and Priority Scheduling for Single Battery Charging Station with Multiple Mobile Robots

Hadrien Roy, Elwan Héry, Ryogo Kubo

Docking Control and Priority Scheduling for Single Battery Charging Station with Multiple Mobile Robots

Hadrien Roy
Department of Electronics and Electrical Engineering
Keio University
Yokohama, Japan
roy.hadrien@kbl.elec.keio.ac.jp

Elwan Héry
Laboratoire des Sciences du Numérique de Nantes (LS2N)
École Centrale de Nantes
Nantes, France
elwan.hery@ec-nantes.fr

Ryogo Kubo
Department of Electronics and Electrical Engineering
Keio University
Yokohama, Japan
kubo@elec.keio.ac.jp

***Abstract*—Interplanetary exploration and colonization are rapidly expanding fields. Robots without self-charging capabilities can be used to reduce the existing heavy payloads on spaceships, and their charging processes can be transferred to a single charging dock. This study proposes the use of vision and navigation ROS2 (Robot Operating System 2) packages to create a docking algorithm for the robots. A communication and queuing system was considered to determine the ranking of each robot that requires charging. Simulations confirmed the effectiveness of the proposed docking algorithm and priority scheduler.**

***Index Terms*—AprilTag, dock, docking manager, charging queue, priority scheduler**

I. Introduction

Robots can be heavy, and they carry a large payload in a spacecraft, thus creating less room for other equipment. To solve this issue, multiple robots can share one docking and charging station, allowing extra payload for other needs. To use a single charger for multiple robots, the robots must manage to organize themselves such that they can all charge flexibly without running out of battery in the field.

The first step is to navigate the robot to the dock. Guangrui *et al.* [1] proposed a two-step vision-based autonomous docking system for a mobile robot using AprilTag with an enhanced ORB-SLAM [2]. AprilTag is a visual fiducial system that can compute the precise 3D position, orientation, and identify the identification number of the tag relative to the camera [3]. We used a docking control system similar to that proposed in [1] which consists of three phases: approach, pose adjustment, and docking. Although this method works well for a single robot, additional states must be added for multiple robots.

When multiple mobile robots are used, wireless communication systems, such as Wi-Fi, must be used. A small solution is ideal, and Cai *et al.* [4] proposed a compact, low-cost radio-frequency system-on-chip method using a microcontroller and a transmission chip. Although this could be an ideal system for transmitting messages between machines, additional hardware and software are required. ROS2 uses data-distribution services (DDS) for communication [5]. The primary mechanism in DDS is called a domain ID which allows for having different logical networks share a physical network [6]. The domain ID is used to compute the user datagram protocol (UDP) ports used for communication and discovery. Every ROS2 node that uses the same ID can freely pass messages between nodes, thus eliminating the need for additional hardware and software.

To charge multiple robots using a single dock, there must be a queue of some sort and possibly a priority system. Several studies proposed different fuzzy-based approaches for prioritizing charging for electric vehicles (EVs). However, they based their successful results only on power conservation and/or charging costs [7], [8], and did not consider the wait time for the EVs. Wait times are important because a robot relies on its power source to move autonomously for task completion; if time is wasted in waiting to charge, the task may not be completed efficiently. Hussain *et al.* [9] proposed an algorithm using a fuzzy inference system to optimize wait times for EVs. This algorithm considers multiple chargers and the same initial starting location for each EV at the charging center. Mobile robots return to charge only when required; therefore, their initial locations vary. Park *et al.* [10] proposed a fuzzy-based scheme to schedule charging of EVs with multiple charging centers. This scheme considers varying the initial distances, charging times, and charging speeds of each EV. For manual operations, the driver determines when charging is required. However, for autonomous operations, the robot determines when charging is required.

The objective of this study was to demonstrate the feasibility of an autonomous docking controller for multiple robots with a singular charging station based on ROS that allows each robot to charge without running out of power before reaching the dock. For this, the following are the requirements:

- A ROS2 fuzzy-based priority ranking and queue system that considers multiple robots, a single charger, and that uses varying initial distances and battery percentage as requirements.
- A docking controller that allows the robots and docking manager (DM) to communicate via services and topics [5]

based on six main states that provide autonomous navigation to the charger and waiting areas. Dock localization, pose adjustment, and self-localization of the robot rely on the use of AprilTags and the ROS Nav2 package [11].
- A docking determination algorithm based on distance, battery percentage, and queue length that calculates when charging is needed.

The paper is organized as follows. Section II describes the docking controller that handles docking determination and the navigation required to reach the charger. The priority scheduler that places the robots in a charging queue when charging is required and handles the fuzzy ranking system is described in Section III. In Section IV, we describe the ROS2 architecture. The simulation results are presented in Section V. Finally, Section VIpresents the conclusions.

II. Docking Control

The system in this study allows multiple robots with a single battery-charging station or dock. The docking control described in this section was applied to each robot individually, which allows them to determine when docking is needed and how to navigate to the dock from its location in the charging queue, which is described further in Section III.

A. Docking Determination

To determine when a robot needs to initiate its docking sequence to charge, various factors such as the distance from the docking station, velocity, battery percentage, and battery voltage can be examined. The robot must ideally initiate its docking sequence when the current battery percentage or voltage required to return to the dock is equal to the current battery percentage or voltage of the robot. However, this would only be ideal if all measurements are perfect and the robot encounters no obstacles or other robots.

The current battery percentage of the robot, P_{curr}, is used to calculate the required battery percentage, P_{req}, needed to return to the dock safely. To account for potential obstacles and the waiting time of other robots in the charging queue, the battery reserve or percentage buffer is subtracted from the required battery percentage as (1):

$$P_{req} = P_{curr} - P_{buff} - Q_{buff}, \tag{1}$$

where P_{buff} is a constant set to a fraction of the maximum percentage, and Q_{buff} is determined by the size of the queue and predetermined charging time. In this study, a short charge time of 60 s was assumed.

The robot must satisfy one of the following conditions to be considered eligible for docking: The first condition depends on battery percentage. Battery dissipation tests were performed to determine the battery percentage required to return to the dock. From the test data, we obtained P_m, which is the battery percentage used per meter at a given velocity. When multiplied by the current distance of the robot D_{curr}, the percentage needed to return to the dock is found by the term in the right side of the inequality in (2)

$$P_{req} - P_{min} \leq D_{curr} \times P_m, \tag{2}$$

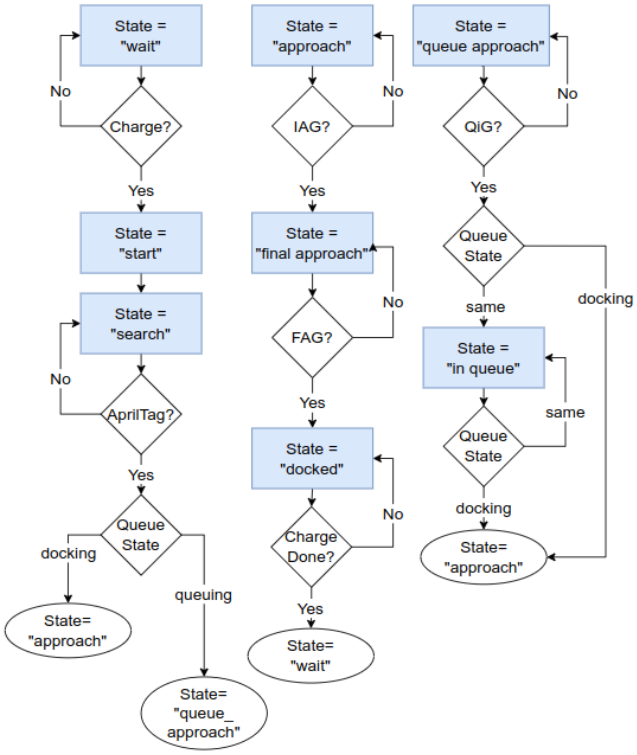

Fig. 1. Simplified docking algorithm control states used to navigate robot from starting point to charger.

where the minimum percentage P_{min} required to drive the motors is subtracted from P_{req}, yielding the actual percentage required based on the motor specifications. This condition is satisfied when the required battery percentage is less than or equal to the percentage required to return to the dock. The second condition is dependent on the distance between the robot and dock. If the current distance D_{curr} of the robot, when subtracted by a distance buffer D_{buff}, is more than half of its maximum allowed distance, this condition is satisfied as in (3).

$$D_{curr} - D_{buff} > \frac{D_{max}}{2}, \tag{3}$$

where D_{buff} is a constant set to account for obstacle avoidance and D_{max} is arbitrarily set depending on the user requirements, which must be within the capabilities of the robots.

If either of these docking eligibility conditions is true, then the robot ceases its current task and begins the docking procedure.

B. Docking Algorithm

The proposed method uses two non-docking states (wait and start), four docking states (searching, approach, final approach, and docked), and two queuing states (queue approach and in queue), as shown in Fig. 1.

Wait State: This is the default docking state in which the robot performs nondocking actions. When the battery needs to be charged, it will move to the "start" state.

Start State: In this state, the motion of the robot is halted, and a request to the DM to add a new robot to the charging queue is made. If accepted, the robot will receive a pose goal or movement command, then enter the "search" state.

Search State: The robot begins its AprilTag detection node. The Python bindings for the AprilTag library were obtained from [12]. While searching for an AprilTag, the ROS2 Nav2 package was used to navigate to the origin of the map. Nav2 is a ROS2 navigation stack that uses behavior trees in combination with different servers to complete various navigation tasks. When an AprilTag is detected, it creates the approach and final approach pose goals. Depending on the queue state of the robot, it can either proceed to the "approach" state or the "queue approach" state.

Queue Approach State: If the queue state of robot is "queuing", it enters the "queue approach" state. The robot then proceeds to its queue-index goal (QiG), which is provided by the DM. Each robot receives an index in the queue that is used to calculate its queue goal. Once it reaches QiG, it changes to the "in queue" state.

In Queue State: Once in this state, the robot sends a request to the DM to change its queue state from "queuing" to "queued." The robot waits in the queue at its designated QiG until the currently charging robot has completed charging. At this point the robot in the queue receives a new QiG to move forward in the queue, or begins docking with the charger if it is next in the queue to charge.

Approach State: The sole function of this state is to navigate towards the approach goal pose while avoiding obstacles using the ROS2 Nav2 package. The approach goal pose is set 1 m in front of the AprilTag as shown in Fig. 2. Once the initial approach goal (IAG) is reached, it will move to the "final approach" state.

Final Approach State: The objective of this state is to appropriately adjust the pose of the robot to align with the dock. It first rotates itself appropriately and then moves forward as required until the final approach goal (FAG) is achieved. Fig. 2 shows the FAG set to 0.5 m in front of the AprilTag or dock. To achieve this goal, the robot should be within the docking limits to complete its charge.

Docked State: When the FAG is reached, the robot moves to this state. In this state, the robot stops its motion and completes charging.

III. Priority Scheduling

The priority scheduler handles the addition of new robots to the charging queue, assigns priority ranks, performs priority determination if needed, and sends docking commands to the robots. The charging queue is structured as a variable queue. However, it functions as a queue for the necessary charging order of the robots and can change when the priority determination is deemed necessary.

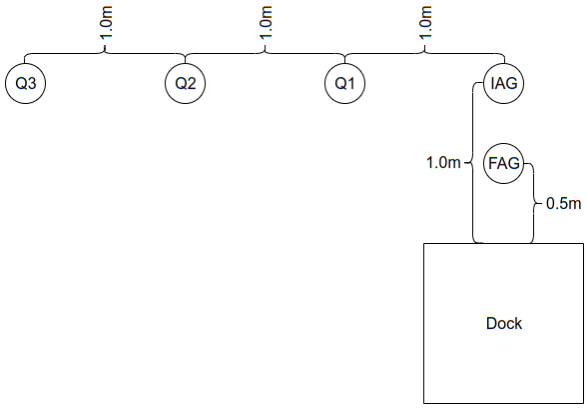

Fig. 2. Layout of dock and docking positions (IAG and FAG are the initial and final approach goals, respectively, and Q1–Q3 are the queue index goals (QiGs)).

When one robot is already docked, the other robots waiting to charge are assigned waiting areas or QiGs, as points beginning with Q, for example, Q1–Q3, as shown in Fig. 2, with the number being the index of the robot in the queue. The first waiting area is 1 m away from the IAG to avoid collision when a robot is charged and for easy access to the goal. Each additional robot waiting area is located 1 m behind the previous one.

A. Priority Scheduler

The priority scheduler has three main functions: adding new robots to the charging queue, changing the queue state of the robots, and sending docking commands to the robots. A robot can request to be added to the charging queue or change its queue state. After these requests are completed, the DM sends new movement commands to all robots in the charging queue via service calls.

1) Adding a New Robot: When a robot requests to be added to the charging queue, it sends its ID, distance from the origin, and battery percentage to the DM. The DM then computes and assigns a rank to the robot based on the provided information. The queue is then checked to determine the queue state for assigning the robot. If the queue is empty, the robot is assigned to the "docking" state (moved to the dock to charge), added to the queue, and prepared for the docking procedure. If the queue is not empty, it is assigned to the "queuing" state (moved to queuing area). Simultaneously, the state of the last robot is checked, and the current robot is added to the queue accordingly. If the last robot in the queue is in a "queuing" state, then a priority ranking is performed, and the appropriate states are reassigned. At this point, the queuing procedure for this request is completed, and the DM sends each robot in the charging queue docking commands based on their state and rank.

2) State Change: Another possible request to a DM is a state change. This occurs when a robot completes an action related to the moving queue state ("docking" or "queuing") and arrives at its destination (FAG or QiG) or when charging is completed. If the current state of a robot during the request

is “docking” or “queuing”, then it is reassigned to “charging” (currently charging) or “queued” (waiting in the queuing area), respectively. If the current state is “charging,” then the robot is removed from the queue, and the next available robot in the queue is assigned to “docking”. At this point, the scheduler finishes the request and sends commands to the robots, if needed.

3) Movement Commands: Once the scheduler completes the queue for the current request, it reassesses the queue and sends each robot an appropriate movement command. A robot can only move after it or another robot is finished charging or when it is initially added to the queue. Once a robot completes charging, the first robot in the queue moves from Q1 to IAG as shown in Fig. 2, performs pose adjustment, and finally moves on to the FAG. If a robot is at Q2, it moves forward in the physical queue to Q1 and the process continues.

B. Priority Queue States

A robot can move through either two or four priority states; only two states, “docking” and “charging”, are used for a robot if the charging queue is empty. The robot enters the “docking” state and navigates to the dock and switches to the “charging” state when it is successfully connected to the charger. If the charging queue is not empty, the robot must first use the “queuing” and “queued” states to wait for the previous robots to complete charging. Although multiple robots can be in the “queuing” or “queued” states, only one maybe be in the “charging” or “docking” states at a time.

C. Priority Ranking

The factors used to determine the priority ranks were the distance from the origin and battery percentage. A shorter distance and battery percentage are given a higher priority. Because a shorter distance has a higher priority than a longer distance, wait times can be optimized. Fuzzy logic is used to obtain the priority rank for each robot. A rule-based table containing nine rules is presented in Fig. 3, which depends on the combination of two different factors: distance and battery percentage. These rules are based on “if-then” rules and were selected for simplicity because rule-based systems are robust.

Each robot receives an initial ranking score when it requests to dock. Using fuzzy logic for a simpler priority comparison, its initial rank is then grouped into one of the five categories: very high, high, medium, low, and very low.

D. Priority Determination

Priority determination is aimed at optimizing the waiting times for charging when multiple robots need to be charged simultaneously. When two or more robots are actively moving to the queuing areas (“queuing” state), a priority determination can be made because the robot requested later may or may not have priority over the previous robot to dock first. One scenario in which this might occur is if Robot 1 is farther and requests first when Robot 2 is closer to the dock. Even though Robot 1 has an earlier request in this case, if Robot 2 docks first, then the overall wait time is reduced because Robot 1 would still be navigating toward the dock. Robots are not compared if they are in the same rank or the rank directly above or below. This can help minimize collisions and unnecessary comparisons.

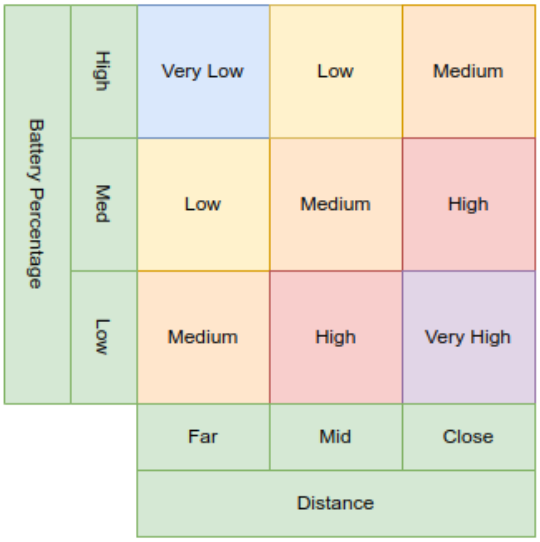

Fig. 3. Fuzzy rule-based table used for assigning robots to a priority rank.

IV. Architecture

This section describes the ROS2 architecture of the system shown in Fig. 4 which comprises multiple robots and a single charging dock.

A. Robot Architecture

1) Docking Client: This node determines whether the robot is eligible for docking by obtaining and using distance and battery data to determine whether it meets the eligibility conditions. If it meets the eligibility conditions, it begins the docking procedure and the AprilTag detection node.

2) AprilTag Detection: The only functions that this node serves are to perform AprilTag detection, pose estimation, and publishing the estimated poses as frames.

3) Docking Controller: This node has the following two main functions: First, it communicates the data back and forth with the DM. The robot sends its ID, distance, and battery percentage to the DM and requests to dock. If the request is accepted, the Docking Controller moves to its second function as the docking state manager. The role of the docking state manager is to dock the robot successfully. To achieve this, we used the docking algorithm described in section II-B.

B. Dock Manager Architecture

The Dock Manager has two functions: granting docking access and maintaining a charging queue with a priority scheduler. When a robot requests to dock, the DM assigns the robot to one of five designations and updates the charging queue. It then sends appropriate movement commands to each robot on the list.

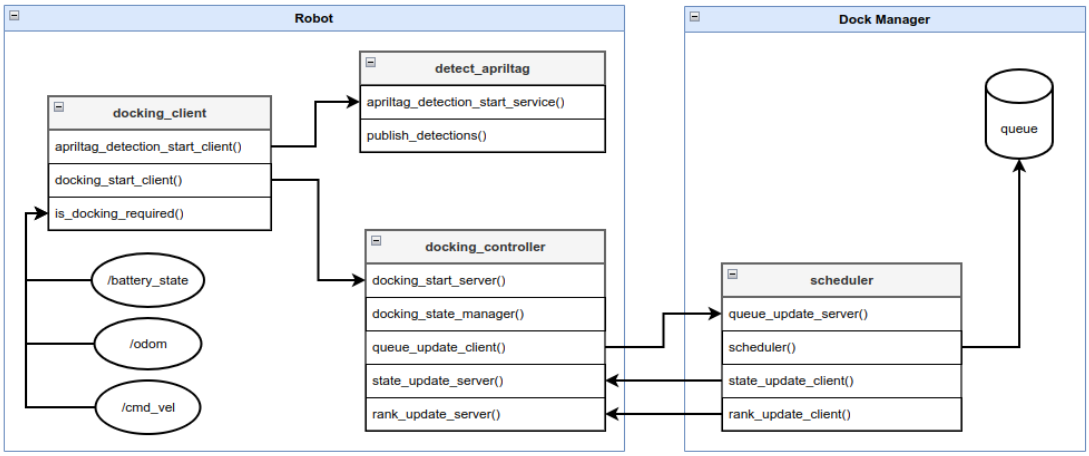

Fig. 4. Simplified ROS2 architecture that includes three nodes for a single robot and one node for the DM.

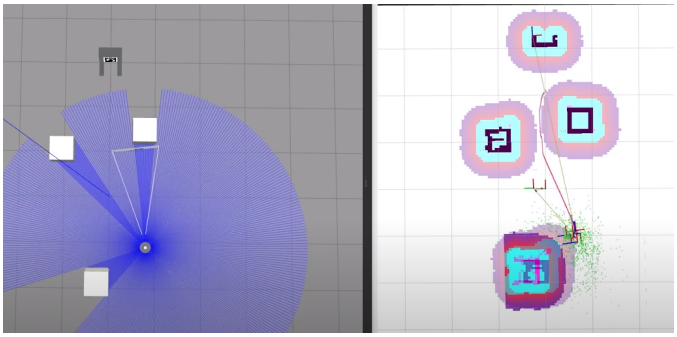

Fig. 5. Simulation of a single robot demonstrating the docking algorithm.

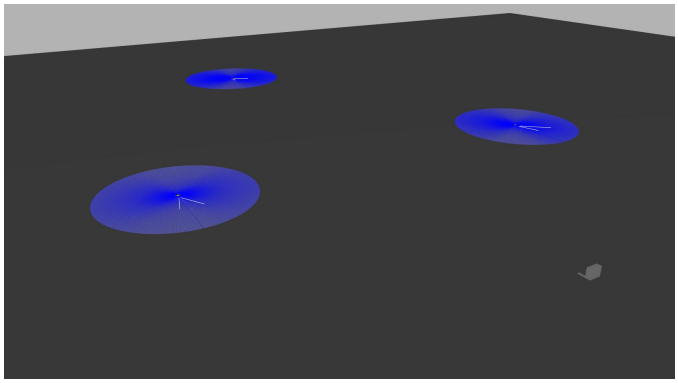

Fig. 6. Simulation of the priority scheduler using three robots and a dock.

V. Simulation and Results

This section shows simulations and their results to confirm the effectiveness of the proposed docking algorithm and priority scheduler. Two Gazebo simulations were conducted in this study. The robot selected for this simulation was the Turtlebot3 Waffle-Pi.

A. Docking Control

Fig. 5 shows the first simulation consisting of a single robot and demonstrates the docking algorithm without a priority scheduler to verify its navigation method. The left side of the figure shows the Gazebo simulation with three randomly placed obstacles (cubes), the robot with its light detection and ranging (LiDAR) field (blue rays) and the grey dock. The right side of the figure illustrates the ROS visualization (RVIZ) client, thereby showing the robot, occupancy grid, detected objects, and planned path. The robot could successfully dock for each of the 10 tests conducted using the Nav2 package, a map for obstacle avoidance, and AprilTag detection for navigation and self-localization.

B. Priority Scheduling

The results of the second simulation are shown in Fig. 6, verifying the functionality of the priority scheduler. It shows the three robots, their LiDAR fields, and the dock. To test this hypothesis, three robots were simulated using three different sets of distance thresholds and one set of battery thresholds, which were used for the fuzzy rankings listed in Table I. The low threshold for the battery begins at 27% because that is the minimum percentage needed to power the motors. Only one battery threshold was tested in order to have less dependent variables during testing. The distance thresholds were selected to be within the maximum distance that a robot can travel, which was arbitrarily set at, 100 m. Each set of thresholds was tested for three percentage buffer constants: 10%, 5%, and 2.5%. Each combination of threshold and percent buffer constants consisted of 18 different tests using combinations of initial positions, as shown in Table II. In the simulation, because the robots did not have any real battery data, a pseudo–battery node was created using a linear timer to reduce the battery percentage. Although this did not accurately mimic a real battery, it allowed us to test the functionality of how the ranks can change over time when the robots move toward the docking area; the docking determination method described in Section II-A. All batteries were initially at 100% charge in these simulations. Because battery data were simulated,

TABLE I
DIFFERENT FUZZY RULES THRESHOLD LIMITS TESTED

Distance Threshold	Close	Mid	Far
1	0–25 m	25–62.5 m	62.5+ m
2	0–10 m	10–25 m	25+ m
3	0–33 m	33–66 m	66+ m

Battery Threshold	Low	Med	High
1	27–50%	50–75%	75–100%

TABLE II
STARTING POSITIONS OF ROBOTS USED FOR SIMULATION IN METERS

Position	Robot 1	Robot 2	Robot 3
1	(−10, 0)	(−10, 10)	(−10, −10)
2	(−30, 0)	(−30, 30)	(−30, −30)
3	(−70, 0)	(−70, 70)	(−70, 70)

navigation was not performed in any of these tests as the docking determination algorithm relied on both accurate battery data and velocity. Therefore, timers were used to simulate movement times while assuming no obstacle avoidance and a constant velocity of 0.1 m/s when navigating. Table III shows the results of all tests conducted. To pass a test, all robots must successfully complete charging before their batteries die. If the battery of a single robot dies before completion, the test is considered a failure. All tests were performed for each threshold set at 10% and 5% percentage buffer passed. Only 94.4% of the tests performed at each threshold set passed the 2.5% percentage buffer. Each failed test at this threshold failed at the same initial position of the robots (one far robot and two close robots). This is because the percentage buffer was not sufficiently high as the queue buffer only updates when a new robot is added to the queue. If too many robots are added simultaneously, the docking determination algorithm does not have sufficient time to increase the queue buffer, thereby producing a failed result.

VI. CONCLUSION

In the current study, a docking algorithm was developed using ROS2 and AprilTags, which was successfully tested using a single-robot simulation. A priority scheduler was created to accurately rank robots in a charging queue and send movement commands to each robot accordingly. The simulation results verified the functionality of the priority scheduler, which could accurately rank robots based on fuzzy ranks, update their states, and dock without running out of power.

To apply the proposed method to interplanetary exploration and colonization, the pseudo–battery node must be adjusted by acknowledging the effects of temperature variation and battery degradation on a robot by being on another planetary body as well as the operating conditions of the robot and its actual battery depletion rate.

To verify the proposed method, the communication using long-distance Wi-Fi needs a further discussion because this

TABLE III
PASSING TEST RESULTS USING THREE DIFFERENT DISTANCE THRESHOLDS SHOWN IN TABLE I AND THREE DIFFERENT PERCENT BUFFER CONSTANTS

Threshold	10%	5%	2.5%
1	100%	100%	94.44%
2	100%	100%	94.44%
3	100%	100%	94.44%

can consume large amounts of energy. Thus, the maximum distance from the dock and maximum number of robots the proposed method can handle should be considered. Once these parameters are known, then the range of the robots can be maximized and possible work area switching can be performed when multiple chargers are used.

Testing physical robots once all simulations are complete is also necessary. Currently, a physical robot has been assembled and its default software has been tested. A camera was installed and calibrated. The code from the simulation was transferred to the physical robot in steps and carefully tested. AprilTag detection was tested, and a tag was successfully detected. The docking algorithm was also tested; however, changes were made to the mapping and odometry features. A local laptop was used to simulate the DM. A priority scheduler was implemented and tested via Wi-Fi for a single robot. We also successfully tested the priority scheduler using service calls on the terminal.

REFERENCES

[1] F. Guangrui and W. Geng, "Vision-based autonomous docking and re-charging system for mobile robot in warehouse environment," in *Proc. 2nd Int. Conf. Robotics and Automation Engineering (ICRAE)*, Dec. 2017, pp. 79–83.

[2] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[3] UMich, [Online]. Available: https://april.eecs.umich.edu/software/apriltag.

[4] G. Cai and Y. Harada, "Design of group robots system based on wireless communication," in *Proc. 2011 Int. Conf. Information Technology, Computer Engineering and Management Sciences*, vol. 2, Sep. 2011, pp. 19–22.

[5] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Sci. Robot.*, vol. 7, no. 66, eabm6074, May 2022.

[6] *Ros 2 documentation*, 2022. [Online]. Available: https://docs.ros.org/en/foxy/index.html.

[7] N. Odkhuu, M. A. Ahmed, and Y.-C. Kim, "Priority determination based on fuzzy logic for charging electric vehicles," in *Proc. Eleventh Int. Conf. Ubiquitous and Future Networks (ICUFN)*, Jul. 2019, pp. 295–299.

[8] L. Yao, Z. Damiran, and W. H. Lim, "A fuzzy logic based charging scheme for electric vehicle parking station," in *Proc. IEEE 16th Int. Conf. Environment and Electrical Engineering (EEEIC)*, Jun. 2016, pp. 1–6.

[9] S. Hussain, Y.-S. Kim, S. Thakur, and J. G. Breslin, "Optimization of waiting time for electric vehicles using a fuzzy inference system," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 15 396–15 407, Sep. 2022.

[10] J. Park, Y. Sim, G. Lee, and D.-H. Cho, "A fuzzy logic based electric vehicle scheduling in smart charging network," in *Proc. 16th IEEE Annu. Consumer Communications & Networking Conf. (CCNC)*, Jan. 2019, pp. 1–6.

[11] *Nav2*, 2020. [Online]. Available: https://navigation.ros.org/index.html.

[12] P. L. GmbH, *Pupil-apriltags: Python bindings for the apriltags3 library*, https://github.com/pupil-labs/apriltags, 2022.