Hindawi
Journal of Robotics
Volume 2022, Article ID 9289625, 24 pages
https://doi.org/10.1155/2022/9289625

Research Article

Design, Implementation, and Performance Evaluation of a Web-Based Multiple Robot Control System

U. U. Samantha Rajapaksha,[1] **Chandimal Jayawardena**,[2] **and Bruce A. MacDonald**[3]

[1]*Department of Information Technology, Faculty of Computing, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka*
[2]*Department of Computer System Engineering, Faculty of Computing, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka*
[3]*Department of Electrical, Computer and Software Engineering, The University of Auckland, Auckland, New Zealand*

Correspondence should be addressed to U. U. Samantha Rajapaksha; samantha.r@sliit.lk

Received 24 March 2022; Accepted 11 May 2022; Published 30 May 2022

Academic Editor: L. Fortuna

Heterogeneous multiple robots are currently being used in smart homes and industries for different purposes. The authors have developed the Web interface to control and interact with multiple robots with autonomous robot registration. The autonomous robot registration engine (RRE) was developed to register all robots with relevant ROS topics. The ROS topic identification algorithm was developed to identify the relevant ROS topics for the publication and the subscription. The Gazebo simulator spawns all robots to interact with a user. The initial experiments were conducted with simple instructions and then changed to manage multiple instructions using a state transition diagram. The number of robots was increased to evaluate the system's performance by measuring the robots' start and stop response time. The authors have conducted experiments to work with the semantic interpretation from the user instruction. The mathematical equations for the delay in response time have been derived by considering each experiment's input given and system characteristics. The Big O representation is used to analyze the running time complexity of algorithms developed. The experiment result indicated that the autonomous robot registration was successful, and the communication performance through the Web decreased gradually with the number of robots registered.

1. Introduction

Autonomous robot registration and control is one of the complex tasks in robotic application development. ROS was developed to improve interoperability and reduce heterogeneous multiple robot programming complexities. ROS is a kind of middleware used by developers in robotic applications to reuse most existing software developed by different researchers. There are different nodes, topics, and message formats for different robots in ROS. An algorithm was developed to find the related topics to control different robots in ROS. Therefore, in our system, the main component is the robot registration engine (RRE), which is developed to register multiple heterogeneous robots by getting all related *rostopics*. The Web interface was developed to interact with robots and users using the ROS bridge server. ROS bridge server worked as an interface between the ROS environment and the Web interface. We have developed different Web interfaces to interact with the user and different types of experiments in our research as described by Web interfaces I to V.

Web interfaces I to IV were developed to work with instructions such as moving the robot to a specific location and working with multiple instructions sequentially. Web interface V was developed to work with instructions with semantics. We have used the Gazebo simulator for our experiments. The robot actions and the initial position were changed with time. Therefore, we have created a schedule for each robot to complete movement or navigation in the experiment with Web interface V. Then, we have identified

the relevant ROS topic in corresponding nodes to subscribe and publish the corresponding command values from the user command. The command publishing engine (CPE) is responsible for publishing the ROS command for each action defined in the given user-level instruction.

Different architectures were used to design the heterogeneous multiple robot system, including centralized, distributed, and hybrid mode [1]. Our solution is based on the centralized server architecture as shown in Figure 1.

We have conducted experiments with Web interfaces I to V with different inputs. The state transition system works with multiple instructions when the user issues several commands sequentially. We have derived the mathematical equations for each experiment for the delay time in response to the inputs and system characteristics. The algorithm's running time was expressed using the Big O notation, representing the time complexity.

The following sections are grouped as follows. Section 2 represents a literature survey with background readings and related research works. The methodology with algorithms and main components of the design are presented in Section 3. The experiments and evaluation of the research project with results are described in Section 4. Finally, Section 5 describes the conclusion with future works.

2. Background Studies

There are many research works that are currently related to heterogeneous multiple robot control and communication. Therefore, we have categorized all background reading as multiple robot controls, Web Interface for robot control, and robot programming and control interface with user instructions.

2.1. Multiple Robot Controls. Some research groups have implemented heterogeneous multiple robot control with the help of a human. Seohyun *et al.* have developed layered architecture to manage and control multiple robots with the intervention of humans. They have designed the interface to separate the autonomous and manual parts. They have proposed architecture to control multiple robots with the human intervention. They have separated the manual part and the mechanical part in this architecture. They have enhanced the multiple robot control with the human intervention [2].

Alberri et al. have developed architecture to connect multi-robot heterogeneous systems with a hierarchical system that is mainly based on the ROS. The layered architecture was used in this development. Lower layers were implemented with C and C++ languages. Complex computations were performed by the upper layer and an intermediate level. They have used three different devices (autonomous quadcopter, autonomous mobile robot, and autonomous vehicle) to complete the testing of the system [3].

A system was developed where personal computers work as servers and robots work as nodes. Again, the hybrid architecture based on ROS with multiple robot systems was used. The server processed all complex computation and visualization, and each node in robots was used to process the real-time tasks [1].

There were many research projects with multiple robots, but our work is unique because of autonomous robot registration with the Web interface, performance evaluation, and heterogeneous robots.

2.2. Web Interface for Robot Control. Costa et al. have introduced a Web-based interface for multiple robot communication using ROS. Two services were implemented named monitor and control. In addition, they have implemented operations as robots move forward, move to the right, move to the left, and move backward. The main contribution was to manage heterogeneous robots by laypeople with the help of ROS [4].

Penmetcha et al. have implemented a system to manage robots that are based on ROS and non-ROS with cloud technologies. The robotic applications were executed with machine learning algorithms based on JavaScript-based libraries. The CPU utilization and latency performance were calculated, and an average latency of 35 milliseconds is achieved. In addition, the innovative cloud was developed using Amazon Web services [5].

Singhal et al. have developed a fleet management system with autonomous mobile robots using a single master and cloud-based configuration. In addition, autonomous navigation was used with a global planner. The authors have identified the critical limitation and issues with cloud robotics [6].

Beetz et al. have developed a service named openEASE to work with the available research based on cloud technology. openEASE is a Web-based knowledge service that robotic researchers can remotely access. The researchers can access semantically annotated data from real-world scenarios [7].

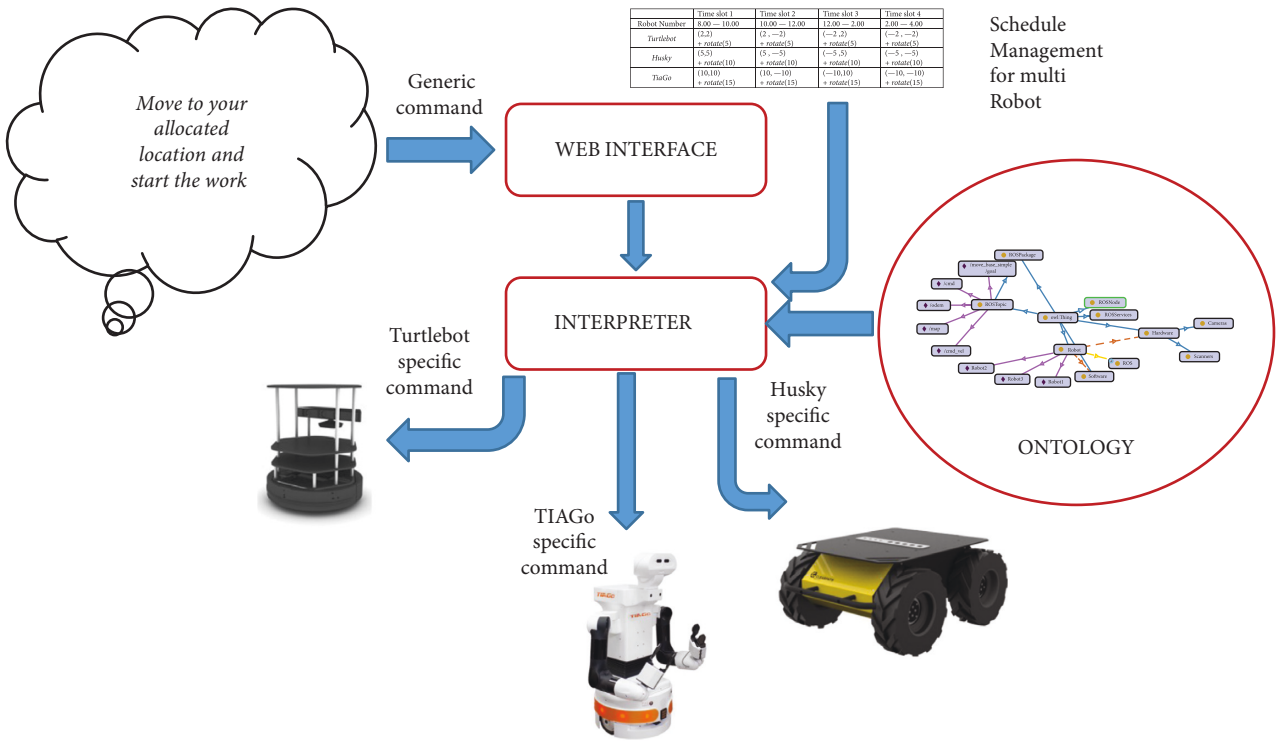
Casañ et al. have implemented a tool with the Web browser interface for online robot programming. It provides the interface with the text box for scripting. MATLAB remote programming environments were used to implement the system [8].

Even though there are many projects with Web interfaces for robot control, our work is different since we have implemented the interface to register and control heterogeneous robots and work with multiple instructions sequentially.

Rajapaksha et al. have implemented a system, which takes user-level instruction with uncertain words for a drone and converts it to machine-understandable executable format using the ontology [9, 10].

Rajapaksha et al. have developed a system to control and communicate with robots using user instruction with uncertain terms. They used the ontology to represent the knowledge of the robot for uncertain terms. The developed system is able to understand the commands such as *go fast* and *go very fast*. They have developed the user-friendly environment to interact with the robots [11, 12].

Rajapaksha et al. have developed a GUI-based system to program and control the robots with Web interface [13].



	Time slot 1	Time slot 2	Time slot 3	Time slot 4
Robot Number	8.00 — 10.00	10.00 — 12.00	12.00 — 2.00	2.00 — 4.00
Turtlebot	(2,2) + *rotate*(5)	(2, −3) + *rotate*(5)	(−2,2) + *rotate*(5)	(−2, −2) + *rotate*(5)
Husky	(5,5) + *rotate*(10)	(5 , −5) + *rotate*(10)	(−5 ,5) + *rotate*(10)	(−5 , −5) + *rotate*(10)
TiaGo	(10,10) + *rotate*(15)	(10, −10) + *rotate*(15)	(−10,10) + *rotate*(15)	(−10, −10) + *rotate*(15)

Figure 1: High-level system diagram.

Rajapaksha et al. have implemented a heterogeneous multiple robot control system by registering robots autonomously with high-level user instructions [14, 15].

Buscarino et al. have proposed a methodology to the control group of robots without central coordination. They have proved that the system performance with having noise can be improved by including long-range connections between the robots. They have modeled the network as a dynamic network [16].

2.3. Robot Programming and Control Interface with User Instructions. Tiddi et al. have developed a system to help nonexpert users in robotics for robotic application development with the help of the ontology in the ROS environment. The main focus was to reduce the time for robot programming for a specific task using the ontology representation. The nonexpert's user needs to configure the system to complete different tasks by the robot [17].

Tiddi et al. have developed the interface, which allows nonexperts to use a robot as a development platform. The system provides high-level commands with the help of fundamental ontology. These ontologies have mapped the high-level capabilities on the robot low-level capabilities (e.g., communication and synchronization). They have used the middleware as ROS [18].

Pomarlan and Bateman have implemented a system that translates "semantic specification" in a natural language instruction to a program that a simulated robot can execute. For example, the system can interpret a sentence into a program that allows the robot to understand the sentence. The main task was to cover a set of basic action concepts from an ontology [19]. Amaratunga et al. have developed an interface to program novel programmer to program easily with interface developed. These ideas can be used for robot programming interface development [20].

Muthugala et al. have reviewed the service robot communications where robots can work with information having uncertainty in natural language instructions. They have implemented the system to identify the issues in working with the qualitative information in the given user instruction in current research work. They have indicated that the quantitative value of information with uncertain terms can depend on the environment, previous experience, and the current context [21].

Sutherland and MacDonald have created domain-specific language to work with the text, which is named as RoboLang. That language is working with the existing programming tools. In addition, the program code can be executed on other robot platforms with minor modification of the code [22].

Datta et al. developed an integrated development environment for visual programming by abstract textual domain-specific language. It provides the program development environment to program robotic applications very fast and very simple with the user requirements [23]. Jayawardena et al. developed a new concept named as coach player model to learn from user commands [24, 25].

Gayashini et al. have developed a navigation model in an unknown area with obstacles. They have developed a reverse navigation model based on previous knowledge [26]. Panagoda et al. have developed a similar system with a potential field graph. They have developed a recovery behavior algorithm to find an alternative path if the current path has any obstacle [27].

Jayawardena et al. have implemented a system to implement software for a given robotic programming scenario within a minimum amount of time. Less coding can be used to create software for the given scenario. The software can be modified, and all changes are made quickly without any errors. The behavior execution engine (BEE) was used to integrate the subsystems together [28].

Datta et al. have developed a system with an environment to develop the program for robots with interactive behaviors. Moreover, it is a visual programming tool. Subject matter experts (SMEs) can involve in the service robot application development. It makes the post-software deployment easy [29].

Kim et al. have developed a system to understand the qualitative information with commands for service robots using the ontology. They have used lexicon semantic pattern matching to get the most relevant keywords from the user instruction. They developed an interpretation system as a prototype, and it was tested with many commands. Standard vocabulary and semantics were defined in the ontology that intelligent agents can use [30].

Scibilia et al. have reviewed motor control theory and sensory feedback applications performed in parallel. Optimal control models were developed to represent the humans' ability to behave optimally after a certain level of training. The advantage of the structural model and Hosman's descriptive model is discussed in this review [31].

Bucolo et al. have worked on a complex and imperfect electromechanical structure that can be used as paradigm for the imperfect system. They have indicated that the electrical and mechanical interactions generate complex patterns because it prevents system to reach correct conditions [32]. Our solution may not be perfect in terms of performance characteristics.

Rashid et al. have developed an algorithm named cluster matching to get the orientation and localization of the robots. Each robot could estimate the relative orientation of neighbor robots that are within its transmission range. It is able to get the absolute positions and orientations of the team robots without knowing the ID of the other robots [33].

Ali et al. have developed the multi-robots navigation model in dynamic environment named shortest distance. The collision-free trajectory was developed using the current orientation and position of the other robots. This algorithm is based on the concept of reciprocal orientation that guarantees smooth trajectories and collision-free paths [34].

According to the above background studies, we can identify that some researches are more similar to our system, but in our system, we have developed an automated robot registration engine that is not available in any other system. Furthermore, our semantic analysis is also based on optimized algorithms compared with the existing techniques used by other researchers.

3. Methodology

The authors have implemented a Web interface to interact with the robots and users. The Web interfaces were developed to interact with different types of experiments in our research. Web interfaces I to IV were developed to work with simple instructions such as moving the robot forward, moving the robot circle, and getting the robot's current position. Web interface V was developed to work with instructions with semantics. We have used the Gazebo simulator for our experiments. The standard ROS JavaScript Library provided by the ROS Web Tools (http://robotwebtools.org/) was used to connect ROS with the Web interface. In the last experiment, the user can issue an instruction like "*Move to the Room 3*" to all robots that are placed at different positions. Figure 2 represents the system architecture of our system.

3.1. Robot Registration Engine. The algorithm that we have developed to register all multiple heterogeneous robots with the human intervention is represented in Figure 3. We have initially created a node called "*regRobot*" to complete the rest of the line execution of the algorithm. IP addresses were extracted from the given IP address list named as "*ipList*." The IP address is used to connect all heterogeneous service robots in the Gazebo environment. Next, ROS commands were executed to collect the software specification, which has used the *execl()* system call by the ROS node created earlier. Finally, an ontology named as "*Registration Ontology*" is created to represent available ROS details.

3.2. Command Interpreter. When a user issues a high-level user instruction on the Web interface provided by the system, the instruction is analyzed by the command interpreter to separate the action, subject, object, and constraint, as shown in Figure 4. First, the instruction can be sent to process the synonyms and semantics. Then, it needs to find out relevant ROS nodes, ROS topics for subscription, and publication with the algorithm as shown in Figure 3.

The system is implemented by handling multiple instructions one by one issued by the user using a state transition diagram with the description of the states as shown in Figure 5. The robot state is saved in the ROS topic to retrieve the robot state from time to time. When the robot is ready, it will accept the user's instruction and complete the assigned work accordingly.

When a user issues multiple instructions to the robot through the Web interface, the related flowchart with the state transition is shown in Figure 6. Initially, a robot must register with the robot registration engine and update the state as ready in the ROS topic. Then, the robot can work according to the instruction given by the user. While the first instruction is processed, the user can issue another instruction and then the robot must be interrupted to handle the second instruction. Based on the priority of the instruction, the robot must be able to decide to continue the current work or start the second instruction. The work state has the highest priority, the motion state has the second highest priority, the dialog state has the third priority, and the ready has the lower priority. Each robot will exit from the system if the instructions are not received within the defined timeout.

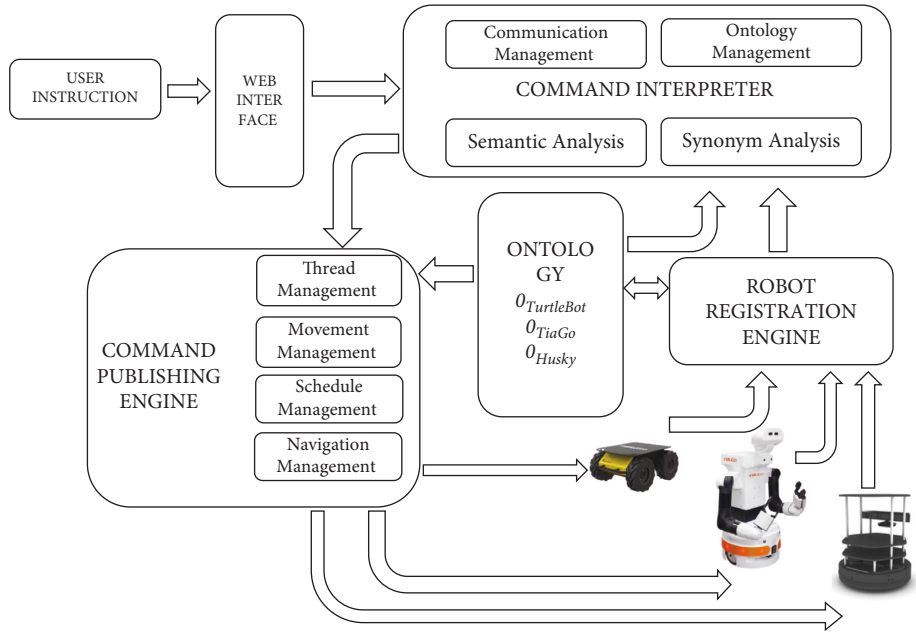


Figure 2: System architecture diagram.

INOUT: *IP address list, URDF file and ontology*

OOTPUT: *Updated ontology with ROS topic, Nodes and services*

ALL_ROBOT_REGISTRATION_ALGORITHM (*ipAddressList, URDF_File, RobotOntology* **)**

1. **Begin**
2. Develop a Process as *"regRobot"* using ROS node//*All lines will be run by this process.*
3. **Foreach** *ipAddress* ∈ *ipAddressList* **do**
4. //*The hardware details were updated with user involvement*
5. Connect the Robot using IP address
6. **If** *Robot has wheels* **Then**
7. **Foreach** $i \in \{1,2,3,4,5,6,7,8\}$ ***do***
8. *Get the hardware details with user and URDF_File*
9. *Update RobotOntology*
10. **Endfor**
11. **Else**
12. **Foreach** $j \in \{bipedal, tripedal, quadrupodel\}$ ***do***
13. *Get the hardware details with user and URDF_File*
14. *Update RobotOntology*
15. **Endfor**
16. **Endif**
17. //*Update the software details with ROS commands*
18. **Foreach** *item* **in** $\{rostopic, rosnode, rosservice\}$ ***do***
19. *Select the necessary option for each command*
20. *Execute item on command prompt with the execl() system call*
21. *Collect the output lists and set as L*
22. **Foreach** *Line* **in** *L* ***do***
23. *Update the ontology with relevant class*
24. **Endfor**
25. **Edfor**
26. **Endfor**
27. *Publish the ontology with "regOnto".*
28. **End**

Figure 3: Robot registration algorithm.

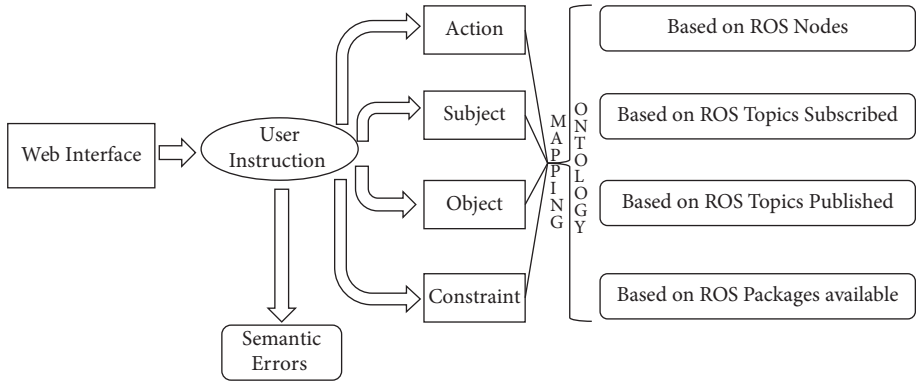


Figure 4: Initial interpretation process.

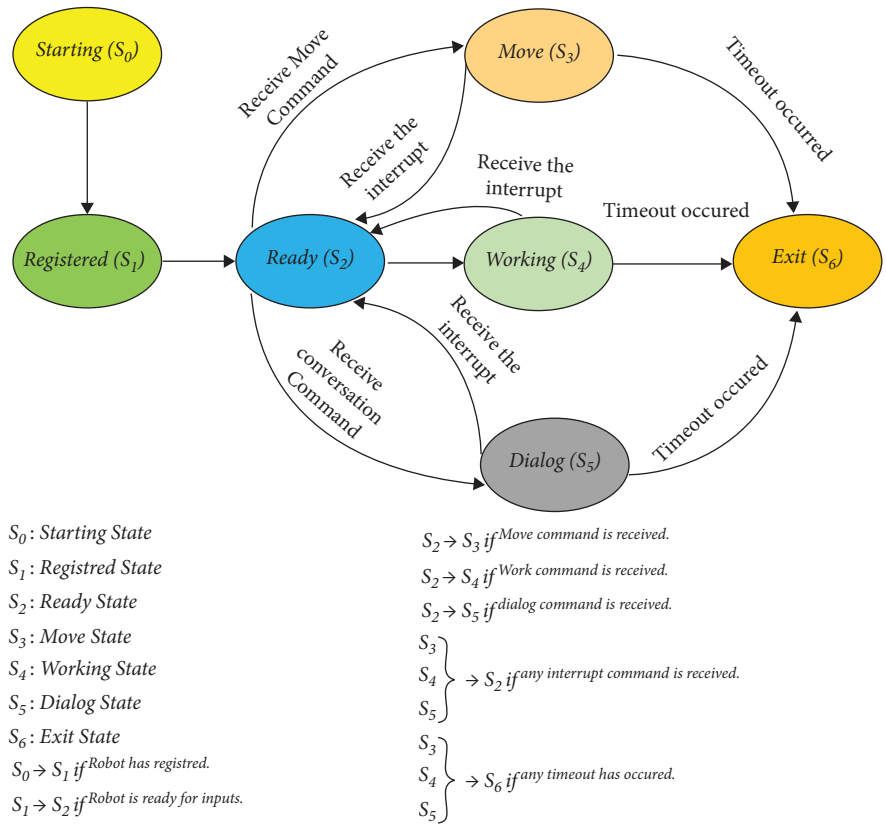


Figure 5: State transition diagram.

3.3. Movement Management. The most critical component of our experiments is the movement of the robots using different instructions using different interfaces. Once a robot is registered with the RRE, it uses the ROS topic identification algorithm to identify the corresponding ROS topic for the movement. In experiment 01, the authors have used teleoperation to move robots forward and circle in an open environment in Gazebo. In experiments 02, 03, and 04, the authors have used the Web-based interface to move robots forward and circle in an open environment in Gazebo with multiple robots. Finally, in experiment 05, the robot was moved to a specific location using the algorithm given in Figure 7. The notations used in the flowchart are described in Table 1.

3.4. Synonym Analysis. Users can enter different types of instructions as described in Table 2 and based on the command interpreter outputs, and the system accepts only commands and commands with the condition. There can be some commands with different verbs with the same meaning, called synonyms. Robots may not be able to understand synonyms until it is appropriately programmed. Therefore, we implemented ontology, which is created with the Web ontology language property called "*sameAs*" to find the synonyms in the given instruction. We have used the "*owl:sameAs*" statement to identify the two uniform resource identifiers, which means each individual has the same "*identity*." We can take the example as synonyms for instruction "*move*" are "*shift, go, proceed, walk, and advance*."

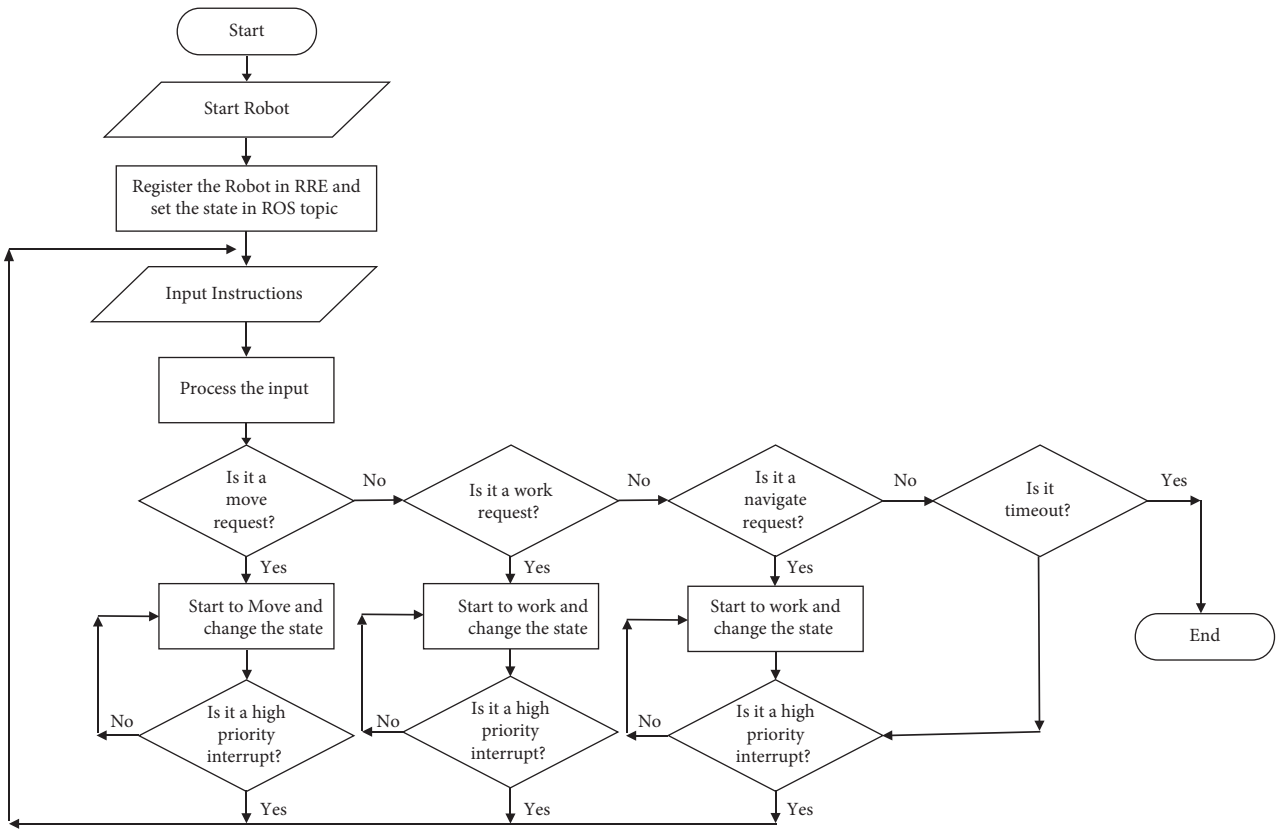


Figure 6: Flowchart for multiple instruction handling.

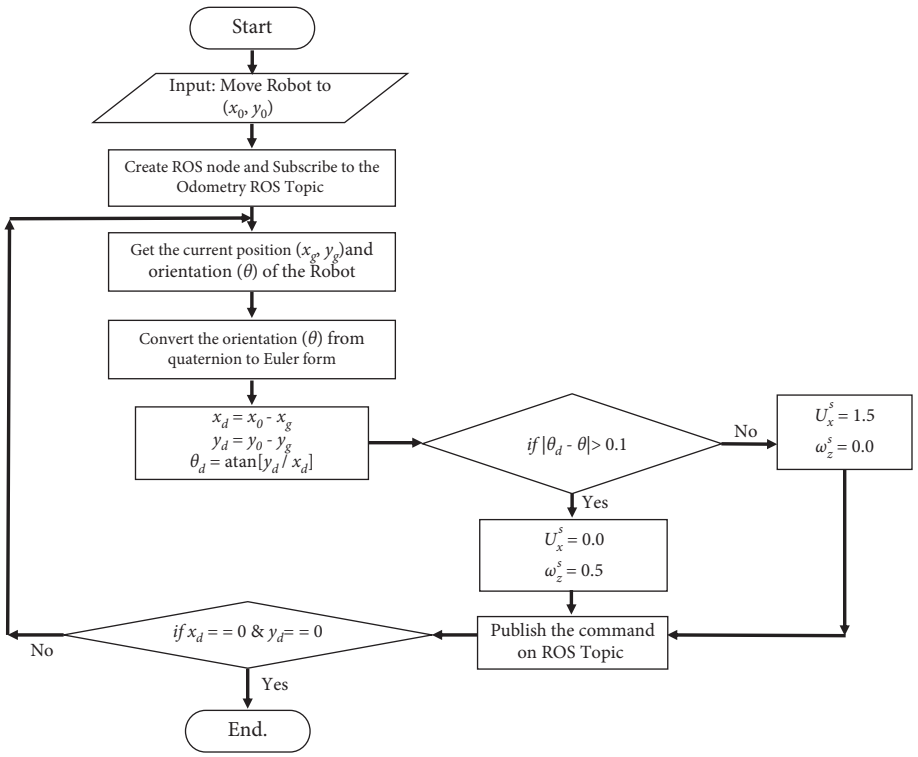


Figure 7: Flowchart for moving robot to a specific goal.

Table 1: Notations used in the flowchart and experiments.

Notation	Description
U_x^s	Linear speed of the robot in x-direction at the start in ms^{-1}
ω_z^s	Angular speed of the robot in z-direction at the start in ms^{-1}
ω_z^e	Angular speed of the robot in z-direction at the stop in ms^{-1}
θ	Current robot orientation in quaternion form
θ_d	The difference between current robot orientation and goal orientation in quaternion form

Table 2: General goal and task scheduling table.

Robot name	Time slot 1 $t_0 - t_1$	Time slot 2 $t_1 - t_2$	Time slot 3 $t_2 - t_3$	Time slot 4 $t_3 - t_4$
R_1	$Goal_{1,1} + Task_{1,1}$	$Goal_{1,2} + Task_{1,2}$	$Goal_{1,3} + Task_{1,3}$	$Goal_{1,4} + Task_{1,4}$
R_2	$Goal_{2,1} + Task_{2,1}$	$Goal_{2,2} + Task_{2,2}$	$Goal_{2,3} + Task_{2,3}$	$Goal_{2,4} + Task_{2,4}$
R_3	$Goal_{3,1} + Task_{3,1}$	$Goal_{3,2} + Task_{3,2}$	$Goal_{3,3} + Task_{3,3}$	$Goal_{3,4} + Task_{3,4}$

Users can update ontology manually. Synonym identification is used in the ROS topic identification algorithm for publishing commands. Different heterogeneous service robots can use different ROS topics; therefore, we need to find the correct ROS topic to publish the commands.

3.5. Semantic Analysis. The semantic meaning of the command is one of the main tasks in interpreting the user-level instructions. Suppose a robot can detect a semantic error in the given user-level instruction that will better implement the robot's intelligence. For example, when a user issues a user-level instruction with the verb "go," we can guarantee that the next part should be a location or destination. The semantic analysis algorithm is described in Figure 8.

The ontology code has a property that requires restricting all robots from moving to a specific position. "*owl:allValuesFrom*" is the property that can be used to define the class with all possible values of the given property defined by "*owl:onProperty*." If the object is not in the restricted value list, it is considered an invalid command and gets the user intervention.

3.6. Ontology. Ontology is a model used to represent the concept and the relationships among all related concepts; for example, if we select the robot's ontology, we can represent all concepts in the robot domain and the relationships among all concepts related to robots [35–37]. Finding concepts from the ontology is the one that takes more time because the running time complexity of the searching algorithm is given by $O(n)$, where n is the number of classes in the given ontology. The part of the ontology that we have created is shown in Figure 9.

3.7. Command Publishing Engine. According to the user-level instruction issued, the command interpreter can identify the action *(move, navigate, identify)* subject, constraint, and object defined in the user instruction. The command publishing engine needs to identify the corresponding ROS topics relevant to the action to publish and subscribe for initiation of the action. For example, if we want to move the robot to a specific location, we can publish the command on ROS topics such as *cm d_vel*, *cm d_vel_mux*, or *cm d_vel_mux/input/navi*. These ROS topics will be varying from robot to robot in heterogeneous environments. The possible ROS topics for the movement and ROS topic for the initial pose are shown in Figure 10.

When a user enters the instruction to all heterogeneous service robots, we need to initiate the action for each robot. This task is completed by command publishing engine (CPE), which can publish the action on the corresponding ROS topic. Initially, CPE can locate the current position of each robot using the optimized algorithm. Get robot position algorithm of each robot is defined in Figure 11. The algorithm has used the *IP* address and the undated ontology to get the initial position and the orientation.

We have created a node in ROS called "*initPos.*" It is responsible for running the remaining lines of the defined algorithm. In addition, this node can find the relevant ROS topics related to the initial position and orientation of the robot.

Each robot may have a different ROS topic to subscribe to and publish for different operations. Therefore, we need to identify these topics before executing any commands on each robot. The ROS topic identification algorithm is described in Figure 12. Initially, the system used the given IP address list and port list to connect with all robots. The ROS topic in the ontology, which the RRE generated previously to create a shared file as *rtList,* is used. Then, it called the Get ROSTopic() algorithm, which is used to get the corresponding ROS topics for each action. This algorithm was used to find the ROS topics for each action defined in the user instruction. For example, if the action is to move the robot from one location to another location, then we need to find the corresponding ROS topic used from the identified list as "*cmd*," "*vel*," "*cmd vel*," "*velocity*," "*speed*," "*travel*," and "*run*." If the identified ROS topics list was not matched with the ROS topics received from the RRE, we called Get Uncertain ROSTopic() to find the ROS topics with synonyms of the action based on the ontology. This algorithm uses the synonyms for the given action to find the corresponding ROS topic. If we can find one, we can use the topic for subscribing or publishing the action; otherwise, we need to get the user input to resolve the problem.

INPUT: *User Instruction, Ontology*

OUTPUT: *Identify the instruction as valid or invalid*

SEMANTIC_ANALYSIS_ALGORITHM (*Instruction, object, RobotOntology* **)**

```
1.  Begin
2.  Develop a process using ROS node as "semAlgo" //This is resposible to run following lines.
3.  Foreach class ∈ ontology do
4.  //Get all classes of the ontology
5.      If action == class then      // Find the class for the action
6.          Foreach restriction ∈ class do
7.              If allValueFrom == object then
8.                  Call Get_ROSTOIC(action)
9.              Else
10.                 Output as invalid request
11.             Endif
12.         Endfor
13.     Else
14.         Get the sameAs List
15.         Foreach i ∈ sameAs List do
16.             If action == i then      // Find the synonym for the action
17.             For each restriction ∈ class do
18.                 If allValueFrom == object then
19.                     Call Get_ROSTOIC(action)
20.                 Else
21.                     Output as invalid request
22.                 Endif
23.             Endfor
24.             Else
25.                 Get the user inputs
26.             Endif
27.         Endfor
28.     Endif
29. Endfor
30. End.
```

Figure 8: Semantic analysis algorithm.

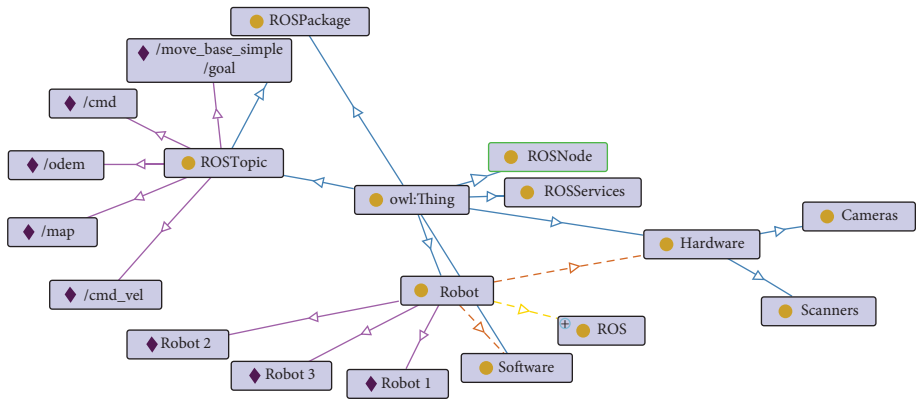


Figure 9: Fragment of the ontology.

3.8. Schedule Management. In our solution, we have assigned scheduled work and location for each robot for a given time slot. The robot can execute user instruction only if it is a free time slot; otherwise, the robot needs to complete the allocated task. The CPE can publish or subscribe to the relevant values for each ROS topic. Each heterogeneous robot has given a specific goal ($G_{i,j}$) or position to move with specific allocated work ($T_{i,j}$) based on the given time allocation as shown in Table 2. According to the given time slot, the location to move (goal) and task to be completed for each robot are displayed in the goal and task scheduling table.

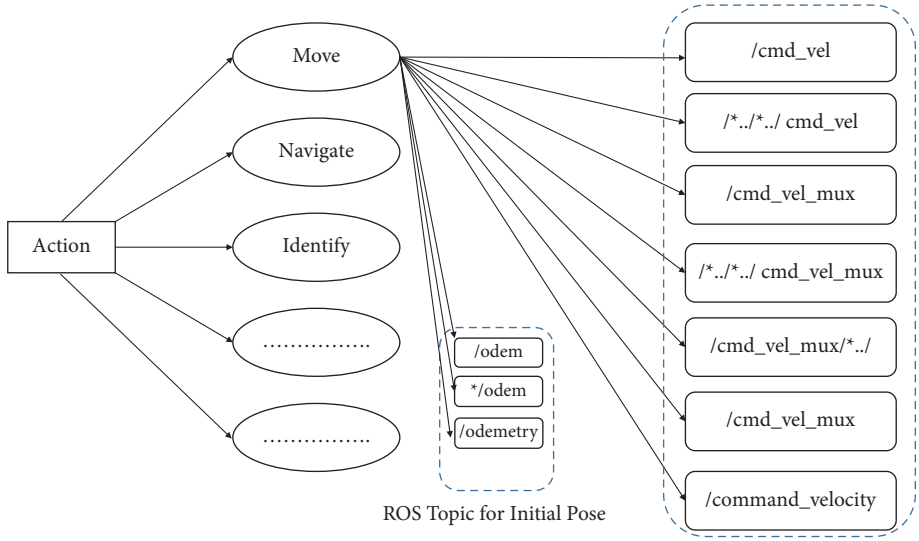


Figure 10: ROS topics for the movement.

INPUT: *Ip List and RobotOntology*

OUTPUT: *Position and Orentation*

Get_Initial_Position_Orientation (*ipList, RobotOntology***)**

1. **Begin**
2. *Develop a process withh node in ROS as "initPos" //This will run following lines.*
3. *Using the ip address connect with the Robot.*
4. *Get the rostopic list from the RobotOntology.*
5. **Foreach** *element* **in** *rostopic list* **do**
6. **If** *element is in {odom,odometry,odomet}* **Then**
7. *Use the element to get odometry content formthe ROS topic*
8. *Retrive the position details in the form of (x, y, z)*
9. *Retrive the Orientation details in the form of (x, y, z, w)*
10. *Add the content in the RobotOntology*
11. **Else**
12. *User needs to provide the ROS topic*
13. *Use the element to get odometry content formthe ROS topic*
14. *Retrive the position details in the form of (x, y, z)*
15. *Retrive the Orientation details in the form of (x, y, z, w)*
16. *Add the content in the RobotOntology*
17. **Endif**
18. *Return the updated ontology.*
19. **Endfor**
20. **End**

Figure 11: Get initial position algorithm.

3.9. Navigation Management. Autonomous navigation of the robot is one of the main research areas in robotic programming. ROS is implemented to work with the navigation stack that is used to navigate from one location to another location easily by hiding most of the complex tasks in autonomous robot navigation. Navigation can be implemented using the *ROS topics*, *message formats,* and *shapes of footprint* of the robot and selecting the relevant

ROS_TOPIC_IDENTIFICATION_ALGORITHM (*ipList*, *portList*)

1. **foreach** $ip_i \in ipList$ & $port_i \in portList$ **do**
2. **Get** the rostopic list form $Ontology_{ip}$
3. **Publish** *ROS Topic as shared file named as* $rtList_i$
4. **foreach** $action_i \in$ {*move, navigate, identify, etc.*} **do**
5. $rt_i \leftarrow$ **GET_ROSTOPIC**$(action_i)$
6. **Record** *the ROS Topics get as* rt_i

GET_ROSTOPIC ($action_i$)

1. Open shared file $f_i \leftarrow rtList_i$
2. **if** $action_i$ = *move* **then**
3. **foreach** $rt \in$ {*'cmd', 'vel', 'cmd_vel', 'velocity', 'speed', 'travel', 'run'*} **do**
4. **foreach** *line* l_i *in file* f_i **do**
5. **if** *rt* **is in** l_i **then** //Exact Matching
6. **Record** *ROS Topic rt for move.*
7. **else** //Non-Exact Matching
8. **GET_UNCERTAIN_ROSTOPIC**($action_i$)
9. **if** $action_i$ = *navigate* **then**
10. **foreach** $rn \in$ {*'move_base', 'map_server', 'robot_amcl'*} **do**
11. **if** *rt* **is in** l_i **then** //Exact Matching
12. **Record** *ROS Topic rn for navigate.*
13. **else** //Non-Exact Matching
14. **GET_UNCERTAIN_ROSTOPIC**($action_i$)
15. **foreach** $rt \in$ {*'cmd', 'vel', 'cmd_vel', 'velocity', 'speed', 'travel',*
16. *'run', 'goal', 'move', 'amcl', 'navigation', 'scan', 'map', 'cloud',*
17. *'rt', 'local', 'global', 'odom'*} **do**
18. **foreach** *line* l_i *in file* f_i **do**
19. **if** *rt* **is in** l_i **then** //Exact Matching
20. **Record** *ROS Topic rt for navigate.*
21. **else** //Non-Exact Matching
22. **GET_UNCERTAIN_ROSTOPIC**($action_i$)

GET_UNCERTAIN_ROSTOPIC (*action*$_i$)

1. *Get the ontology* (O_a) *developed for* $action_i$
2. **foreach** *class* $\in \{O_a\}$ **do**
3. **If** *actioni* == *class* **then**
4. *Get the synonym list* SL_a *from the ontology form the class*
5. **foreach** $rt \in \{SL_a\}$ **do**
6. **foreach** *line* l_i *in file* f_i **do**
7. **if** *rt* **is in** l_i **then** //Exact Matching
8. **Record** *ROS Topic rt for navigate.*
9. **else** //Non-Exact Matching
10. *Get the User Inputs for* $action_i$

Figure 12: ROS topic identification algorithm.

values for the ROS topics for each robot. Odometry and sensor information were used as main inputs for the ROS navigational stack, and then, it generated the corresponding velocity for the mobile base. According to the ROS specification, we can find that the mobile base is controlled by *xisvelocity*, *yisvelocity*, *an dt hetaisvelocity*, and a 2D planner laser is mounted on the mobile base. The navigation is exactly successful on the square-shaped robots.

The map server was used to store the created map file. All heterogeneous service robots used the map stored in the map server to navigate obstacles from one location to another. *amcl* (Adaptive Monte Carlo Localization) file and *move_base* file for each robot were maintained as launch files to localize and move the robot in the given environment. For example, *ROSscan*, *ROSo do metry*, *ROSinitialpose*, and *ROSparticleclou d* topics were used in the *amcl* launch file for each robot for the localization. For example, *ROStopiccm d_vel*, *ROStopicgoal*, *ROStopico de m*, *ROStopiclocal_plan*, *ROStopicglobal_plan*, and *ROStopic f oot print* were used for remapping the *ROStopicmove_base* node for each robot.

3.10. Thread Management. Since we need to control and coordinate multiple robots simultaneously, threads can be used to complete the task efficiently. Furthermore, a thread is a lightweight process inside a process. Therefore, concurrency can be developed using the threads quickly.

4. Experiment and Results

We have conducted the experiments with Web interfaces I to V for simple instructions and measured the response time of

Table 3: Single robot average start/stop response time without Web interface.

	$U_x^s = 0.5\ \text{ms}^{-1}$	$U_x^s = 1.0\ \text{ms}^{-1}$	$U_x^s = 1.5\ \text{ms}^{-1}$
$StartResponse(s)$			
$\omega_z^s = 0.0\,\text{ms}^{-1}$	0.871	0.807	0.787
$\omega_z^s = 0.5\,\text{ms}^{-1}$	0.657	0.541	0.531
$\omega_z^s = 1.0\,\text{ms}^{-1}$	0.561	0.512	0.499
$\omega_z^s = 1.5\,\text{ms}^{-1}$	0.511	0.501	0.476
$StopResponse(s)$			
$\omega_z^e = 0.0\,\text{ms}^{-1}$	1.211	1.728	2.161
$\omega_z^e = 0.5\,\text{ms}^{-1}$	1.039	1.631	1.981
$\omega_z^e = 1.0\,\text{ms}^{-1}$	1.001	1.431	1.871
$\omega_z^e = 1.5\,\text{ms}^{-1}$	0.988	1.181	1.761

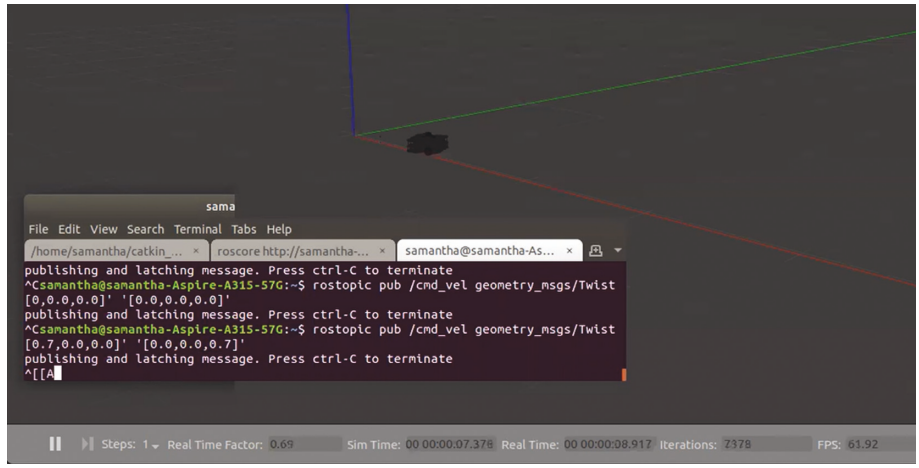


Figure 13: Single robot interaction without Web interface.

the robot start and stop with the Web interface. The initial experiment was conducted without the Web interface. We have used the following notation for our experiments as shown in Table 1.

4.1. Experiment 01: Single Robot Interaction with Simple Instruction without Using the Web Interface. Initially, the authors completed the experiment with a single robot without using the Web interface in the Gazebo simulator with TurtleBot3. The authors have issued instructions to move the robot forward and move in a circle using the terminal interface with the rostopic pub command. We have evaluated the average response time of the robot for a start and stop instructions. We have conducted the experiments with different linear and angular speeds of the robot for start and stop instructions. The experiment results will be displayed as shown in Table 3. The interaction with TurtleBot3 with the terminal without using a Web interface is shown in Figure 13. The response delay for the start and stop of the robot is represented by equations (1) and (2), where $R_{s,d}^{start}$ and $R_{s,d}^{stop}$ represent the single robot delay at start and stop, respectively, $\tau_{d,os}$ represents the delay in system call execution in operating system, $\tau_{d,ROS}$ is used to represent the delay in communicating with ROS topics, and c_1, c_2 are constants.

$$R_{s,d}^{start} = \tau_{d,os} + \tau_{d,ROS} + \frac{c_1}{\{U_x^s + \omega_z^s\}}, \tag{1}$$

$$R_{s,d}^{stop} = \tau_{d,os} + \tau_{d,ROS} + c_2\{U_x^s + \omega_z^s\}. \tag{2}$$

Figure 14 represents the average start and stop response time for the robot for each instruction. The average start response time gradually decreases when the linear and angular speed increases, while the average stop time increases when the linear and angular speed increases.

4.2. Experiment 02: Single Robot Interaction with Simple Instruction with Web Interface without Autonomous Robot Registration. The authors developed the Web interface to interact with the robot using the ROS bridge server. The authors have issued instructions to move the robot forward and move in a circle using the buttons provided in the Web interface with the robot. We have evaluated the average response time of the robot for a start and stop instructions. We have conducted the experiments with different linear and angular speeds of the robot for start and stop instructions. The experiment results will be displayed as shown in Table 4. The interaction with TurtleBot3 with the terminal with Web interface is shown in Figure 15. The response delay

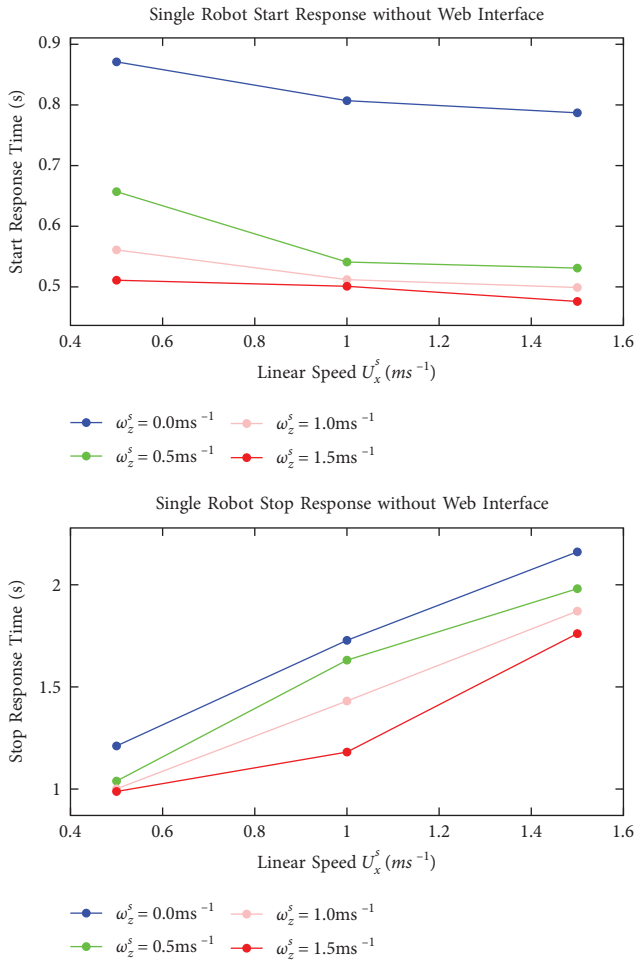


Figure 14: Single robot interaction without Web interface.

for the start and stop of the robot is represented by equations (3) and (4), where $R_{s,d}^{start}$ and $R_{s,d}^{stop}$ represent the single robot delay at start and stop, respectively, $\tau_{d,web}$ represents the delay in communication through Web interface, $\tau_{d,ROS}$ is used to represent the delay in communicating with ROS topics, and c_1, c_2 are constants.

$$R_{s,d}^{start} = \tau_{d,web} + \tau_{d,ROS} + \frac{c_1}{\{U_x^s + \omega_z^s\}}, \quad (3)$$

$$R_{s,d}^{stop} = \tau_{d,web} + \tau_{d,ROS} + c_2\{U_x^s + \omega_z^s\}. \quad (4)$$

Figure 16 represents the average start and stop response time for the robot for each instruction. The average start response time gradually decreases when the linear and angular speed increases, while the average stop time increases when the linear and angular speed increases. According to the analysis, the authors have identified that Web communication is slightly faster than communication through the terminal.

4.3. Experiment 03: Single Robot Interaction with Simple Instruction with a Web Interface with Autonomous Robot Registration. The robot registration engine was developed to collect all robot details, including all ROS topics necessary to subscribe and publish. The ROS topic identification algorithm was developed to select the relevant ROS topics for each action defined in the user instruction. We have evaluated the average response time of the robot for a start and stop instructions. We have conducted the experiments with different linear and angular speeds of the robot for start and stop instructions. The experiment results will be displayed as shown in Table 5. The interaction with TurtleBot3 with the terminal with Web interface is shown in Figure 17. The response delay for the start and stop of the robot is represented by equations 5and 6, where $R_{s,d}^{start}$ and $R_{s,d}^{stop}$ represent the single robot delay at start and stop, respectively, $\tau_{d,web}$ represents the delay in communication through Web interface, $\tau_{d,ROS}$ is used to represent the delay in communicating with ROS topics, $\tau_{d,RT}$ represents the delay in ROS topic identification, and c_1 and c_2 are constants.

Table 4: Single robot average start/stop response time with Web interface.

	$U_x^s = 0.5$ ms^{-1}	$U_x^s = 1.0$ ms^{-1}	$U_x^s = 1.5$ ms^{-1}
StartResponse (*s*)			
ω_z^s = 0.0 ms^{-1}	0.811	0.789	0.766
ω_z^s = 0.5 ms^{-1}	0.753	0.732	0.699
ω_z^s = 1.0 ms^{-1}	0.611	0.601	0.544
ω_z^s = 1.5 ms^{-1}	0.571	0.577	0.501
StopResponse (*s*)			
ω_z^e = 0.0 ms^{-1}	1.031	1.402	1.981
ω_z^e = 0.5 ms^{-1}	1.001	1.267	1.812
ω_z^e = 1.0 ms^{-1}	0.981	1.101	1.602
ω_z^e = 1.5 ms^{-1}	0.911	0.999	1.201

$$R_{s,d}^{start} = \tau_{d,web} + \tau_{d,ROS} + \tau_{d,RT} + \frac{c_1}{\{U_x^s + \omega_z^s\}}, \quad (5)$$

$$R_{s,d}^{stop} = \tau_{d,web} + \tau_{d,ROS} + \tau_{d,RT} + c_2\{U_x^s + \omega_z^s\}. \quad (6)$$

Figure 18 represents the average start and stop response time for the robot for each instruction. The average start response time gradually decreases when the linear and angular speed increases, while the average stop time increases when the linear and angular speed increases. According to the analysis, authors have identified that autonomous robot communication is slightly slower than communication through the Web without autonomous registration.

4.4. Experiment 04: Homogeneous Multiple Robot Interaction with Simple Instruction with a Web Interface with Autonomous Robot Registration. The authors have developed the launch file to create multiple robots in the same Gazebo environment. Initially, two TurtleBot robots were spawned in the empty Gazebo world at two different locations. The simple move instructions were issued to both robots simultaneously and evaluated the average response time for the start and stop instructions. The separate namespaces were used to identify each ROS topic for each robot. The first

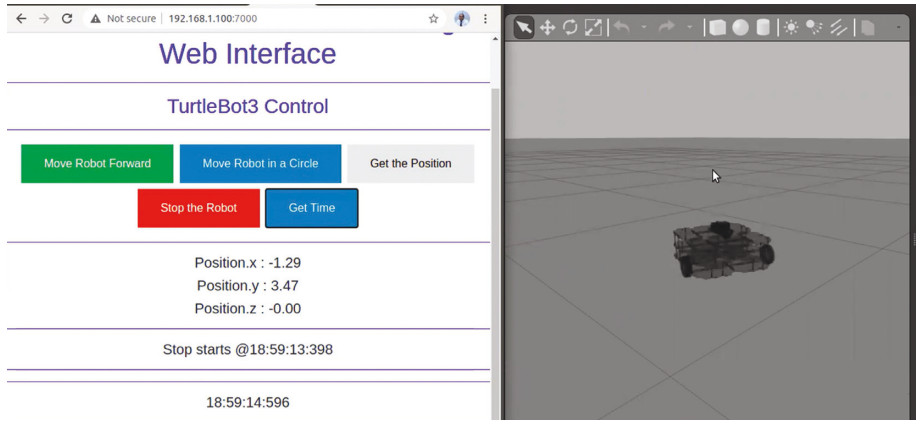


Figure 15: Single robot interaction with Web interface.

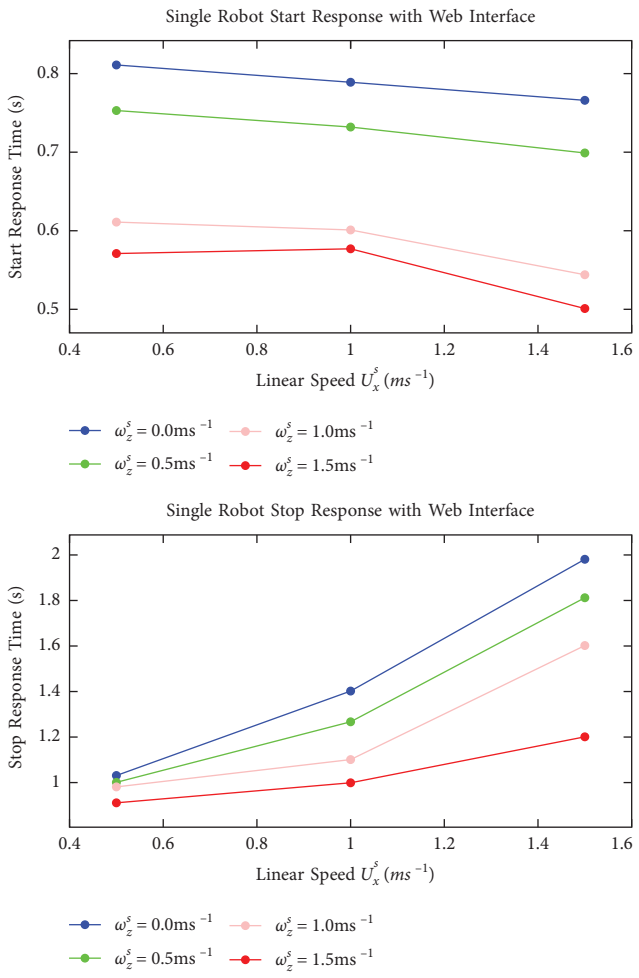


Figure 16: Single robot interaction with Web interface.

robot was named robot 1, and the second one was named robot 2. The interaction with multiple two TurtleBot with the terminal with Web interface is shown in Figure 19. The response delay for the start and stop of the robot is represented by equations (7) and (8), where $R_{m,d}^{start}$ and $R_{m,d}^{stop}$ represent the multiple robots' delay at start and stop, respectively, $\tau_{d,web}$ represents the delay in communication

Table 5: Single robot average start/stop response time with Web interface autonomous.

	$U_x^s = 0.5\ ms^{-1}$	$U_x^s = 1.0\ ms^{-1}$	$U_x^s = 1.5\ ms^{-1}$
StartResponse (s)			
$\omega_z^s = 0.0\ ms^{-1}$	1.011	1.001	0.981
$\omega_z^s = 0.5\ ms^{-1}$	1.001	0.987	0.956
$\omega_z^s = 1.0\ ms^{-1}$	0.987	0.872	0.789
$\omega_z^s = 1.5\ ms^{-1}$	0.861	0.761	0.712
StopResponse (s)			
$\omega_z^e = 0.0\ ms^{-1}$	1.345	1.765	2.552
$\omega_z^e = 0.5\ ms^{-1}$	1.241	1.451	2.222
$\omega_z^e = 1.0\ ms^{-1}$	1.109	1.431	1.988
$\omega_z^e = 1.5\ ms^{-1}$	1.011	1.344	1.765

through Web interface, $\tau_{d,ROS}$ is used to represent the delay in communicating with ROS topics, $\tau_{d,RT}$ represents the delay in ROS topic identification, and c_1, c_2, α, and β are constants.

$$R_{m,d}^{start} = \alpha\left\{\tau_{d,web} + \tau_{d,ROS} + \tau_{d,RT}\right\} + \frac{c_1}{\left\{U_x^s + \omega_z^s\right\}}, \quad (7)$$

$$R_{m,d}^{stop} = \beta\left\{\tau_{d,web} + \tau_{d,ROS} + \tau_{d,RT}\right\} + c_2\left\{U_x^s + \omega_z^s\right\}. \quad (8)$$

Secondly, the authors have spawned another four robots in the same Gazebo environment for the experiment. Separate namespaces were given for each robot to avoid conflicts with the same ROS topic. The simple move instructions were issued to both robots simultaneously, and the average response time for the start and stop instructions is evaluated. The experiment results will be displayed as shown in Table 6. The interaction with multiple four TurtleBot with the terminal with Web interface is shown in Figure 20.

Figure 21 represents the average start and stop response time for the single robot, two robots, and four robots for each instruction where the linear speed is changed, but the angular speed is kept constant to avoid the collision among the robots. The average start response time gradually increases when the number of robots increases, while the average stop time increases when the number of robots increases.

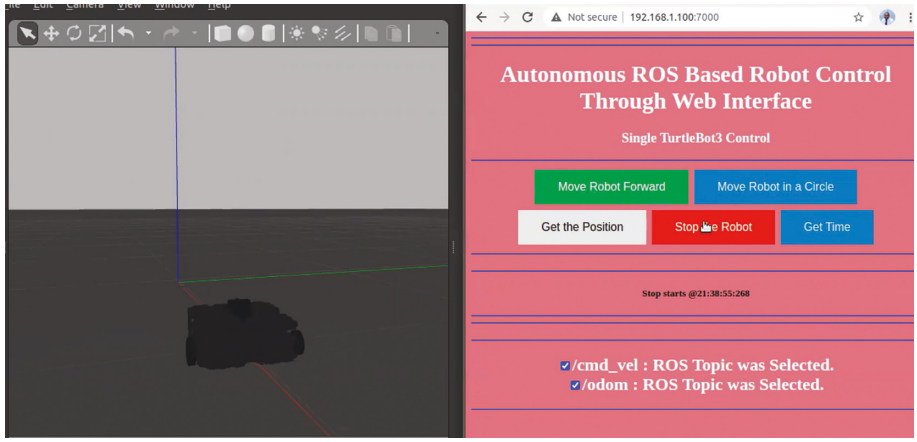


Figure 17: Single robot interaction with Web interface auto-registration.

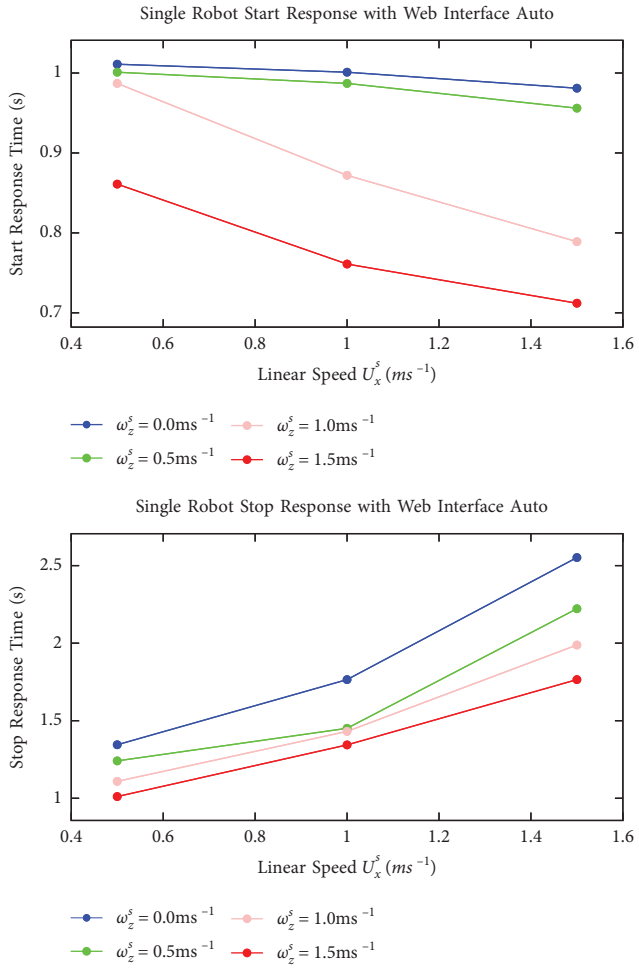


Figure 18: Single robot interaction with Web interface auto-registration.

4.5. Experiment 05: Move the Robots to a Specific Location with a Web Interface with Autonomous Robot Registration. The authors have completed the experiment to move the robot (single robot, two robots, and four robots) to a given target location by an instruction using the Web interface. The robots were placed at different positions to move the same distance on average. The following map represents the initial position and target locations of two and four robots as shown in Figure 22.

The authors have conducted the experiments with a single robot, two robots, and four robots with a single instruction to move the robot to a specific location given by (x, y) coordinates. The average time taken by robots to a specific location was measured and presented in Table 7. The average move time increases with the number of robots and distance, as shown in Figure 23. The delay for moving single robot and multiple robots is represented by equations (9) and (10), where $R_{s,d}^{move}$ and $R_{m,d}^{move}$ represent the single and multiple robots' delay in moving to specific location, respectively, $\tau_{d,web}$ represents the delay in communication through Web interface, $\tau_{d,ROS}$ is used to represent the delay in communicating with ROS topics, $\tau_{d,RT}$ represents the delay in ROS topic identification, $\tau_{d,pos}$ is used to represent delay in getting the current position and orientation of the robot, and c_1, c_2, α, and β are constants.

$$R_{s,d}^{move} = \tau_{d,web} + \tau_{d,ROS} + \tau_{d,RT} + \tau_{d,pos} + \frac{c_1}{\{U_x^s + \omega_z^s\}}, \quad (9)$$

$$R_{m,d}^{move} = \beta\{\tau_{d,web} + \tau_{d,ROS} + \tau_{d,RT} + \tau_{d,pos}\} + c_2\{U_x^s + \omega_z^s\}. \quad (10)$$

4.6. Experiment 06: Robot Interaction with Multiple Instructions with a Web Interface with Autonomous Robot Registration. We have completed the experiment with the multiple instructions issued by the user sequentially with the state transition diagram. The sample interaction between the user instruction through the Web interface and the robot is shown in Figure 24. This diagram represents only three user instructions that the user issues to control the robot. The experiment was conducted with three instructions to move the robot to three different locations. The target locations were represented as (x_0, y_0), (x_1, y_1), and (x_2, y_2). These target locations were selected to make sure all robots move at equal distance on average.

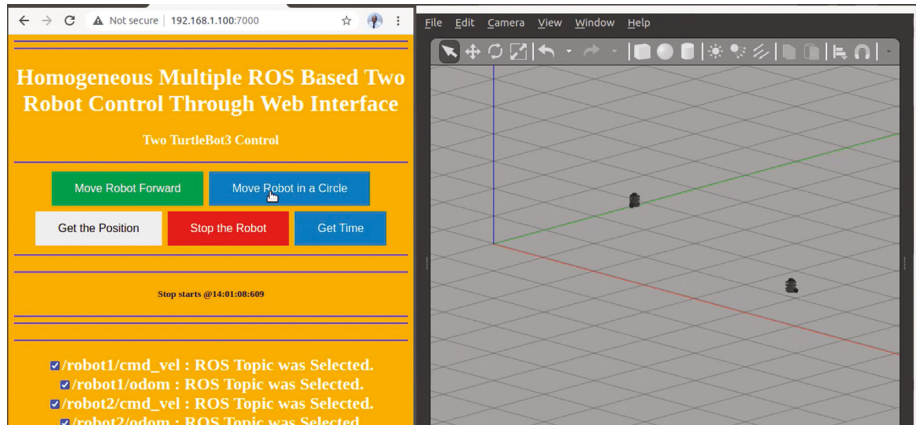


Figure 19: Multiple two robots' interaction with Web interface auto-registration.

Table 6: Multiple robots average start/stop response time with Web interface autonomous.

	$U_x^s = 0.5\ \text{ms}^{-1}$	$U_x^s = 1.0\ \text{ms}^{-1}$	$U_x^s = 1.5\ \text{ms}^{-1}$
StartResponse(*s*)			
SingleRobot	1.011	1.001	0.981
TwoRobots	1.129	1.078	1.016
FourRobots	1.456	1.241	1.112
StopResponse(*s*)			
SingleRobot	1.345	1.765	2.552
TwoRobots	1.674	1.987	2.987
FourRobots	1.987	2.134	2.456

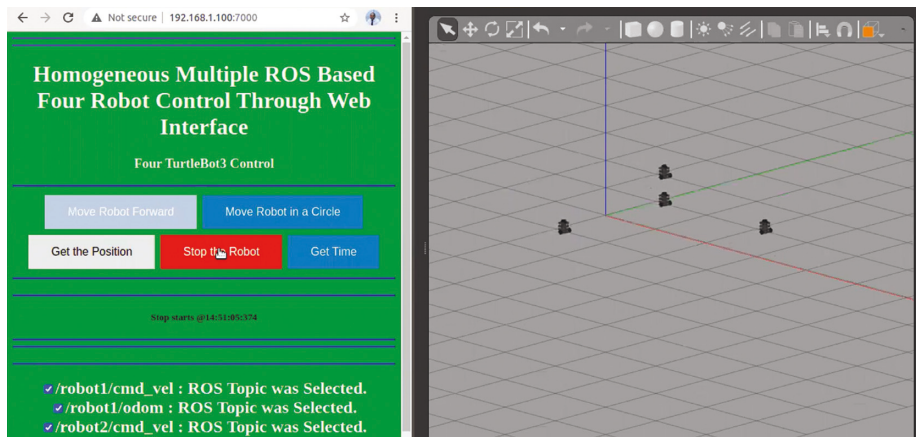


Figure 20: Multiple four robots' interaction with Web interface auto-registration.

The initial robot positions for two robots and four robots are represented in the map given in Figure 25. The robots were initially placed concerning the target locations where each robot must move the same distance. The blue color circle represents the initial robot position. The green color square represents target locations given by user instructions. The target locations are identified to ensure all robots travel equal distances on average.

The equation that represents the delay occurs because multiple instructions issued by user were developed using the mathematical notation. We have used δ_{ij} as state transition time from i to $j, \forall (i, j) \in \{1, 2, 3, 4, 5, 6\}$, S_δ as time taken to save the state in ROS topic, R_δ as time taken to retrieve the state from ROS topic, and ϵ_n as transition delay by n instructions, where $n \in \{1, 2, 3, \ldots, l\}$. The total state transition delay time ϵ_n^s for single instruction $n = 1$ is shown in equation (11). The total state transition delay time ϵ_n^m for multiple instructions $n = 1, 2, 3, ..l$ is shown in equation (12). The delay for moving single robot and multiple robots to specific location with multiple instructions sequentially is

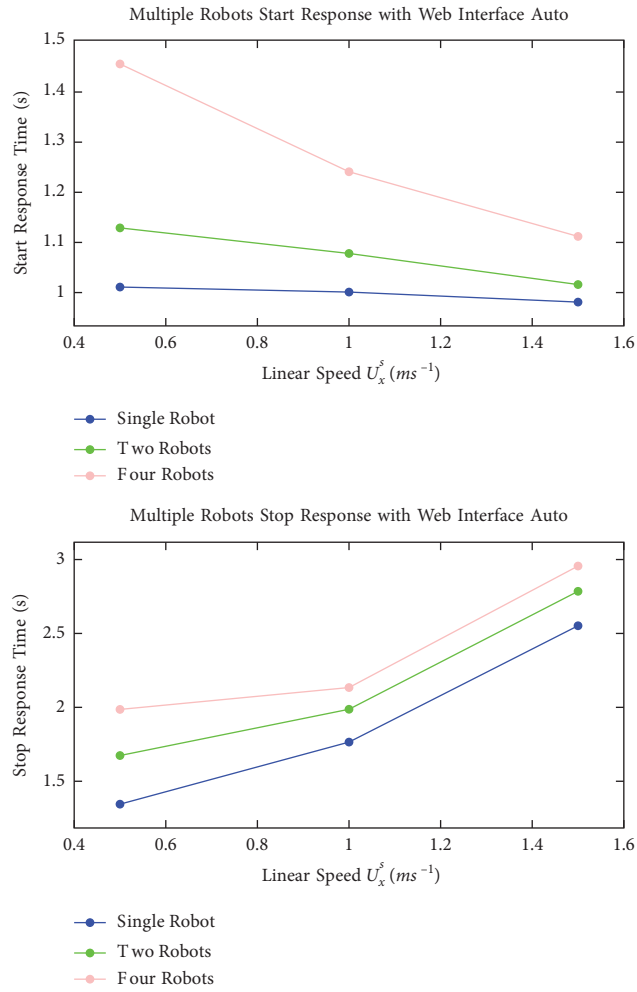


Figure 21: Multi-robot interaction with Web interface.

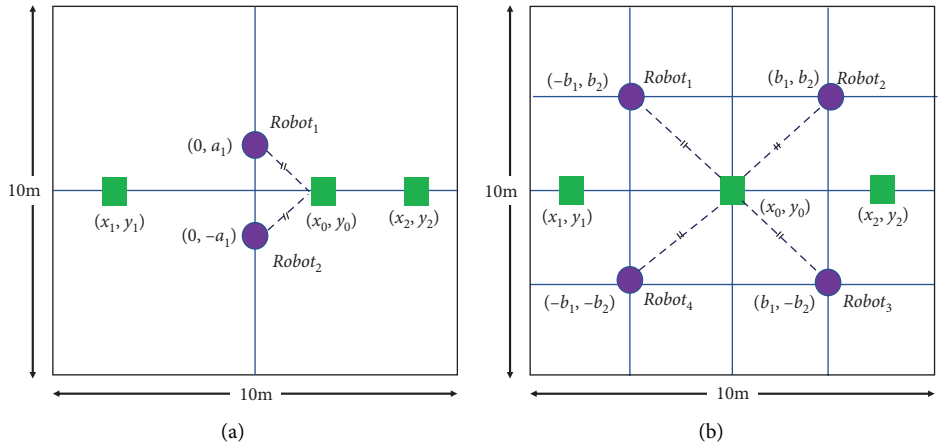


Figure 22: Initial position and target locations: (a) two robots and (b) four robots.

Table 7: Average moving time for multiple robots with single instruction.

Average move time (s)	Move to (x_0, y_0)	Move to (x_0, y_0) and (x_1, y_1)	Move to (x_0, y_0), (x_1, y_1), and (x_2, y_2)
SingleRobot	2.01	2.22	3.01
TwoRobots	2.24	3.01	3.34
FourRobots	3.05	3.21	4.01

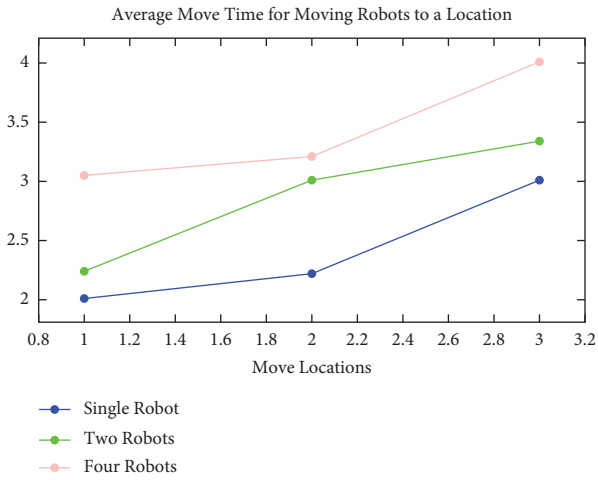


Figure 23: Average move time for moving a robot to a specific location.

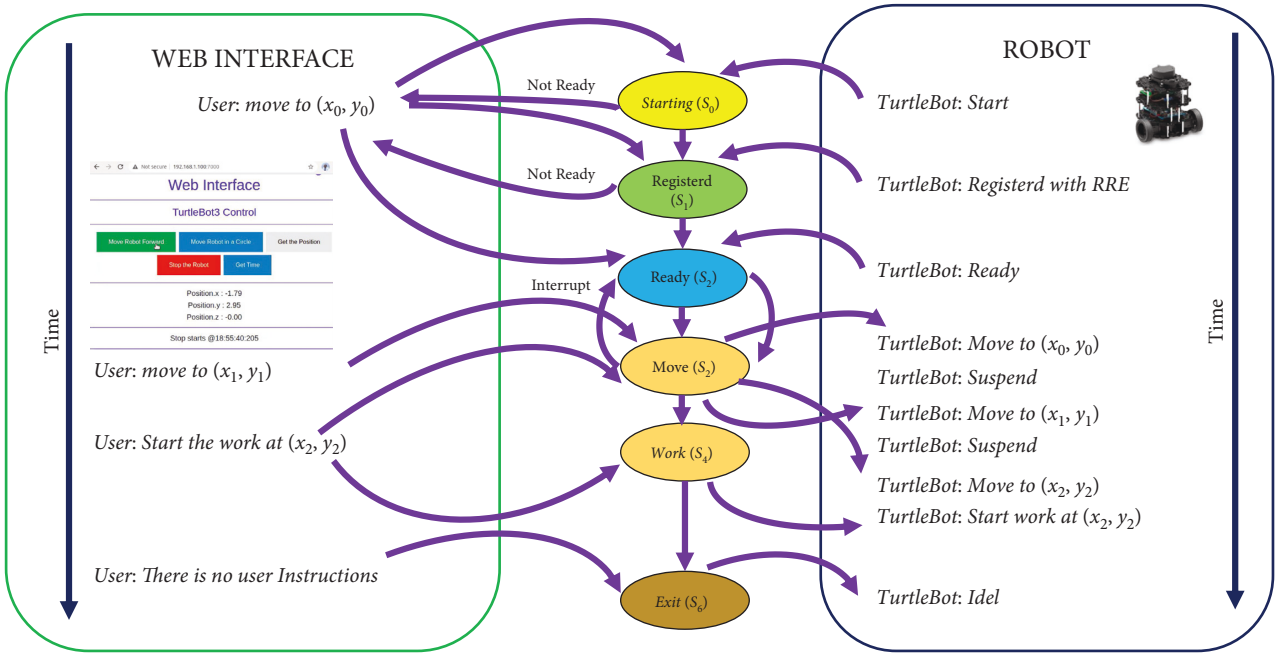


Figure 24: Multiple instructions and robot interaction.

represented by equations (13) and (14), where $R_{s,d}^{mIns}$ and $R_{m,d}^{mIns}$ represent the single and multiple robots' delay in moving to specific location, respectively.

$$\epsilon_n^s = \delta_{01} + \delta_{12} + \delta_{2j} + \delta_{j6} + 7S_\delta + 7R_\delta, \quad \forall j \in \{3,4,5\}, \tag{11}$$

$$\epsilon_n^m = \delta_{01} + \delta_{12} + \sum_{n=1}^{n=l} \delta_{2j} + \delta_{j6} + (3+l)(S_\delta + R_\delta), \quad \text{if } n = l, \forall j \in \{3,4,5\}, \tag{12}$$

$$R_{s,d}^{mIns} = \epsilon_n^s + \tau_{d,web} + \tau_{d,ROS} + \tau_{d,RT} + \tau_{d,pos} + \frac{c_1}{\{U_x^s + \omega_z^s\}}, \tag{13}$$

$$R_{m,d}^{mIns} = \beta\{\epsilon_n^m + \tau_{d,web} + \tau_{d,ROS} + \tau_{d,RT} + \tau_{d,pos}\} + c_2\{U_x^s + \omega_z^s\}. \tag{14}$$

The experiment was conducted with multiple instructions with single, two, and four robots. All robots were given the target locations in each instruction to travel the same distance on average to make the completion time for the comparison. The average completion time is tabled as shown in Table 8. The average completion time and the number of instruction relationships are shown in Figure 26.

4.7. Experiment 07: Heterogeneous Multiple Robot Interaction with Semantic Instruction with a Web Interface with Autonomous Robot Registration. We have evaluated our system in the Gazebo environment using three robots such as turtlebot, husky, and TiaGo. The virtual environment, available in Python httpserver (Python–mhttp), was executed to implement necessary Web pages with JavaScripts

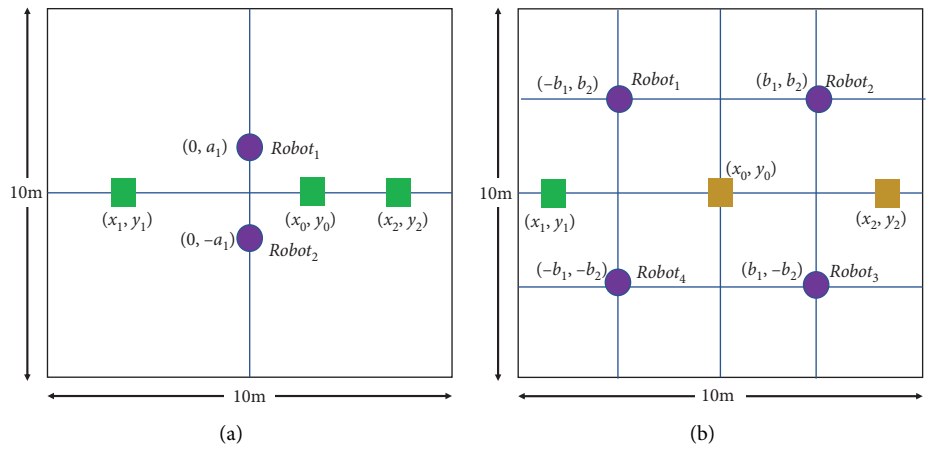


Figure 25: (a) Initial positions of two robots: (b) initial positions of four robots.

Table 8: Average completion time for multiple robots with multiple instructions.

Average competition time (s)	Single instruction	Two instructions	Three instructions
SingleRobot	2.32	2.98	3.56
TwoRobots	2.59	3.24	3.98
FourRobots	3.23	3.57	4.62

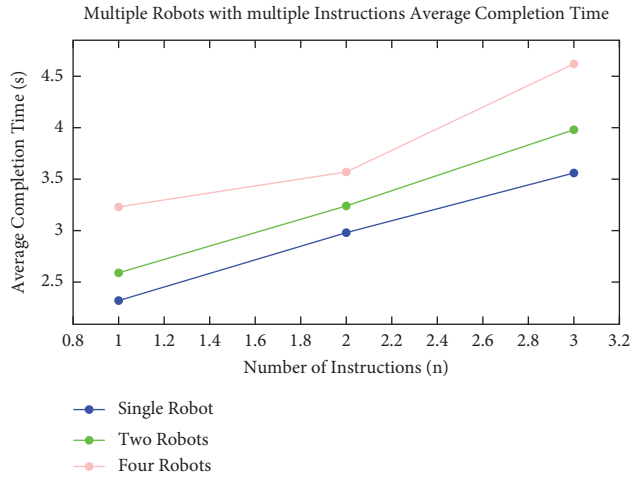


Figure 26: Average completion time with multiple instructions with state transition.

Table 9: Instruction types used for testing.

Instruction type	Description	Example
Type I	Instruction without synonym or semantic issue	Move to A and clean
Type II	Instruction with synonym	Shift to B and clean
Type III	Instruction with semantic issue	Move to roof and clean
Type IV	Instruction with synonym and semantic issue	Shift to sky and clean
Type V	Instruction with synonym and semantic issue (not programmed where user involvement is needed)	Proceed to sea and clean

for the Web interface. We have used the rosbridge server to work as an interface between ROS and non-ROS clients. The user has added the instruction on the Web interface provided by the system to interact with the multiple robots. The instruction types, which were used to test our system, are shown in Table 9. Type I was a general instruction with no synonym or semantic issue. The synonym was added to instruction type II, where a synonym analysis algorithm

Table 10: Time complexity of algorithms.

Algorithm name	Time complexity in Big O notation
Robot Registration Algorithm()	$O(n^4)$
Synonym Analysis Algorithm()	$O(n^2)$
Semantic Analysis Algorithm()	$O(n^3)$
Get Position and Orientation Algorithm()	$O(n)$
ROS Topic Identification Algorithm()	$O(n^4)$

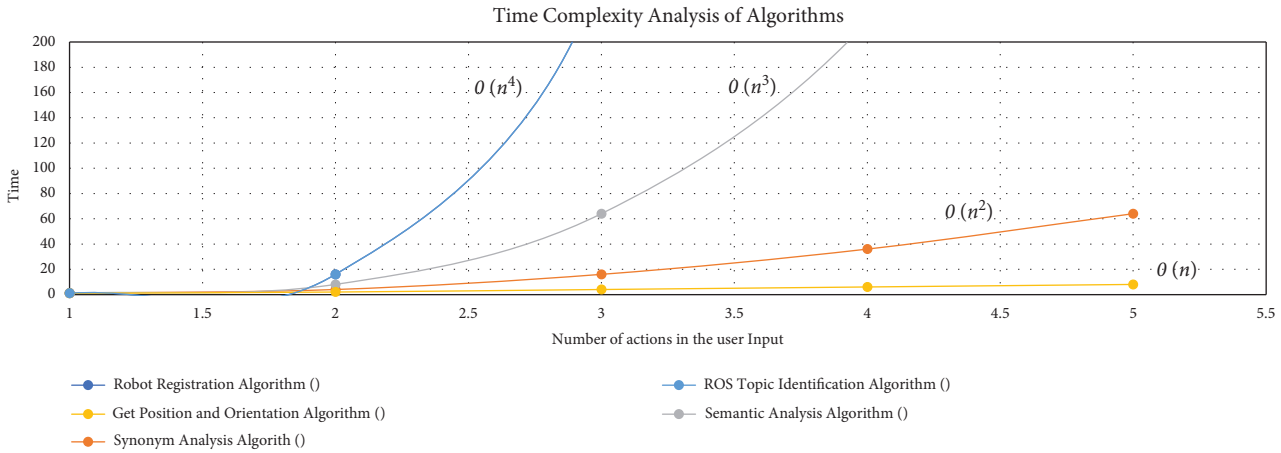


Figure 27: Graph of the time complexity of all algorithms.

Table 11: Instruction types with time complexity.

Types	Algorithms used in command interpreter	Time complexity	Algorithms used in robot registration and command publishing engine	Time complexity
Type I	Analysis algorithm is not needed	$O(1)$	RR Algorithm()+ ROS TI Algorithm()	$O(n^4)$
Type II	Synonym Analysis Algorithm()	$O(n^2)$	RR Algorithm()+ ROS TI Algorithm()	$O(n^4)$
Type III	Semantic Analysis Algorithm()	$O(n^3)$	RR Algorithm()+ ROS TI Algorithm()	$O(n^4)$
Type IV	Synonym Analysis Algorithm()+ Semantic Analysis Algorithm()	$O(n^3)$	RR Algorithm()+ ROS TI Algorithm()	$O(n^4)$
Type V	Synonym Analysis Algorithm() + Synonym Analysis Algorithm()+human intervention is needed	$O(n^3)$	RR Algorithm()+ ROS TI Algorithm()	$O(n^4)$

processed it. The semantic of the instruction is not clear in instruction type III. Instruction type IV has both synonym and semantic issues. The synonym and semantic were not programmed for the instruction type V, where the user has to handle the synonym and semantic issues. The system was tested with many instructions, type I to type V.

The identification of the synonym and the semantic issues were performed by our algorithms accurately. Furthermore, we have completed the time complexity analysis of our algorithm to measure the system's performance using the Big O notation. The time complexities of all algorithms are shown in Table 10. Time complexity is calculated using the number of loops used by each algorithm, where n is the input size. The graph of the time complexity for all algorithms is shown in Figure 27. According to the time complexity analysis, we can identify that the robot registration algorithm and ROS topic identification algorithm have poor performance because time complexity is $O(n^4)$.

Time complexity analysis with Big O notation for each type of instruction is shown in Table 11. Command interpreter has used the Synonym Analysis Algorithm(), and Semantic Analysis algorithm(), where Synonym Analysis Algorithm() has taken $O(n^2)$, and Semantic Analysis algorithm() has taken $O(n^3)$ running time based on the asymptotic notation in algorithm analysis. Therefore, instruction type II is poor compared with instruction type III. Instruction type V is worse because user interaction is needed to solve the synonym and semantic issue in the instruction since synonym and semantics are not programmed.

In addition to the above discussed time complexity analysis for instruction types I to V, we have conducted two types of experiments with the Gazebo environment with *Turtlebot*, *Husky,* and *TiaGo* robots. In the first experiment type, we have moved all heterogeneous robots to a given goal in the open world in the Gazebo, and the second type of experiment is to navigate all heterogeneous robots to a given

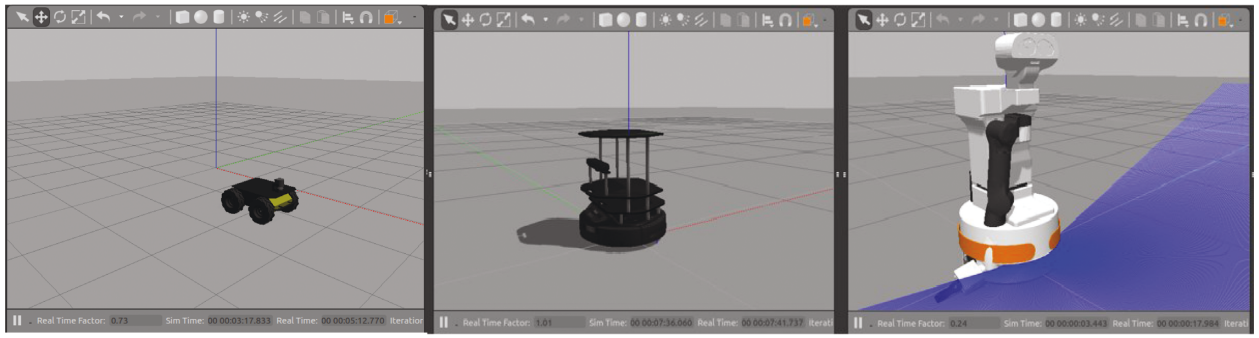


Figure 28: Husky, Turtlebot, and TiaGo robots in empty world.

Table 12: Goal and task scheduling table.

Robot name	Time slot 1 $t_0 - t_1$	Time slot 2 $t_1 - t_2$	Time slot 3 $t_2 - t_3$	Time slot 4 $t_3 - t_4$
Turtlebot	$A(2,2) + rotate(5)$	*FreeTime*	$B(-2,2) + rotate(5)$	$C(2,-2) + rotate(5)$
Husky	$D(5,5) + rotate(10)$	$E(5,-5) + rotate(10)$	*FreeTime*	$F(0,5) + rotate(10)$
TiaGo	*FreeTime*	$G(1,-1) + rotate(15)$	$H(0,1) + rotate(15)$	$I(-1,1) + rotate(15)$

Table 13: Experiment results for goal without navigation.

Robot	Goal without navigation			
Experiment	Goal 01 success rate 08.00-10.00	Goal 02 success rate 10.00-12.00	Goal 03 success rate 12.00-02.00	Goal 04 success rate 02.00-04.00
Turtlebot	0.65	0.85	0.90	0.95
Husky	0.50	0.65	0.70	0.80
TiaGo	0.45	0.55	0.65	0.85

goal with obstacles in the Gazebo. All three robots (turtlebot, husky, and Tiago) in an open world in the Gazebo are shown in Figure 28. Experiments were conducted using the system above multiple robots with movement and navigation using 20 type IV instructions. Users can update the goal and task assigned for each robot for the different schedules in Table 12. We have added the self-rotation for each robot to simulate the task completed by robots based on the scheduled task. We found some errors in robot registration algorithm and ROS Topic Identification Algorithm() for movements and navigation. There were more ROS topic settings than the robot's movement in an open world in navigation.

The results of the experiment are represented in the table for three robots *Turtlebot*, *Husky,* and *TiaGo,* where we have tested 20 times for each goal at 4 different time slots as 8.00-10.00 am, 10.00-12.00 noon, 12.00-2.00pm, and 2.00-4.00pm. We received different ontology searching errors, robot registration errors, ROS topic identification errors, and command publishing errors in each time slot. Therefore, we gradually minimized the error with the experienced we had in each experiment with the timing. The success rate is measured with 20 tests. It defines the number of successful tests without errors out of 20 tests for each robot in each type of experiment.

The results of experiment type 01 (without navigation) are shown in Table 13. According to the analysis, we have identified that the *turtlebot* has a higher success rate compared with other robots, as shown in Figure 29.

The results of the experiment type 02 (with navigation) are shown in Table 14. The success rate is also increasing as similar to experiment 01 as shown in Figure 30.

The running time of the robot registration algorithm and ROS topic identification algorithm is $O(n^4)$, where n is the number of actions defined in the user instruction. These two algorithms had the highest time complexity compared with other algorithms developed in our system.

In general, delay in response time for the start has decreased when the linear and angular speed is increased. However, delay in response time for the stop has increased when the linear and angular speed is increased. Delay has occurred when the robot is controlled without the Web interface because of the delay with system call execution through operating system and delay with communication with ROS functions. When a robot is controlled through the Web without auto-registration, the delay has occurred in communication through the Web and communication with ROS through the ROS bridge server. When the auto-registration was added to the system, then we need to add the delay taken by the algorithm for the ROS topic identification. It is obvious that the delay time increases with the number of robots increased. When the robot is sent to a specific location, then we need to add time taken to get the current position and orientation for the delay time. When a robot is

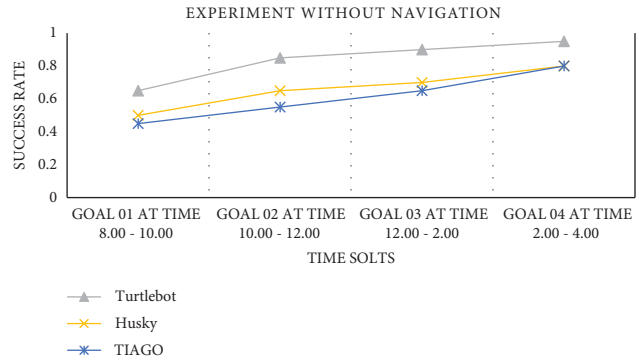


Figure 29: Experiment without navigation success rate.

Table 14: Experiment results for goal with navigation.

Robot	Goal with navigation			
Experiment	Goal 01 success rate 08.00-10.00	Goal 02 success rate 10.00-12.00	Goal 03 success rate 12.00-02.00	Goal 04 success rate 02.00-04.00
Turtlebot	0.40	0.55	0.75	0.80
Husky	0.35	0.40	0.55	0.70
TiaGo	0.30	0.45	0.60	0.75

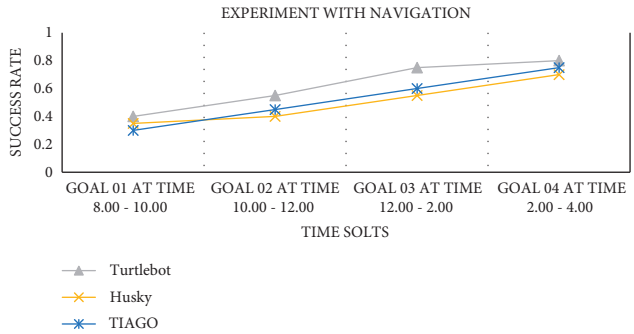


Figure 30: Experiment with navigation success rate.

controlled by the multiple instructions, then we had to use a state transition system. Therefore, we need to add the time taken by the state transition system to save and retrieve the state to the delay time to get the more accurate results. According to the analysis, the authors have identified that Web communication is slightly faster than communication through the terminal.

5. Conclusion and Future Works

This research study has developed a system to issue instruction through the Web interface and controls multiple robots. Initially, all multiple robots need to register with robot registration engine. The autonomous robot registration and autonomous ROS topic identification algorithms were implemented successfully. The delay time is increased with the introduction of these algorithms. We have derived the mathematical equations for each delay time, which varies based on the inputs and system characteristics. The experiment result indicated that the autonomous robot registration was successful, and the communication performance through the Web decreased gradually with the number of robots registered. The running time of the robot registration algorithm and ROS topic identification algorithm is $O(n^4)$. We have not implemented the access control of the multiple robots in the same environment. We will be implementing access controlling and synchronization with all robots in our future work.

Data Availability

There are no data involved in this research.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Sri Lanka Institute of Information Technology under grant number FGSR/RG/FC/2021/05. The authors thank SLIIT (Sri Lanka Institute of Information Technology) for the support given towards this Research Project.

References

[1] C. Hu, C. Hu, and D. He, "A new ros-based hybrid architecture for heterogeneous multi-robot systems," in *Proceedings of the The 27th Chinese Control and Decision Conference (2015 CCDC)*, pp. 4721–4726, Qingdao, China, May 2015.

[2] S. Jeon, M. Jang, and D. Lee, "Control architecture for heterogeneous multiple robots with human-in-the-loop," in *Proceedings of the 2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pp. 274–278, Jeju Island, Korea, November 2012.

[3] M. Alberri, S. Hegazy, and M. Badra, "Generic Ros-based architecture for heterogeneous multi-autonomous systems development," in *Proceedings of the 2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pp. 1–6, Madrid, Spain, September 2018.

[4] L. F. Costa and L. M. G. Gonçalves, "Roboserv: a ros based approach towards providing heterogeneous robots as a service," in *Proceedings of the 2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR)*, pp. 169–174, Recife, Brazil, October 2016.

[5] M. Penmetcha, S. Sundar Kannan, and B. C. Min, "Smart cloud: scalable cloud robotic architecture for web-powered multi-robot applications," in *Proceedings of the 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2397–2402, Toronto, ON, Canada, October 2020.

[6] A. Singhal, P. Pallav, and N. Kejriwal, "Managing a fleet of autonomous mobile robots (amr) using cloud robotics platform," in *Proceedings of the 2017 European Conference on Mobile Robots (ECMR)*, pp. 1–6, Paris, France, September 2017.

[7] M. Beetz, D. Beßler, and J. Winkler, "Open robotics research using web-based knowledge services," in *Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5380–5387, Stockholm, Sweden, May 2016.

[8] G. A. Casañ, E. Cervera, and A. A. Moughlbay, "Ros-based online robot programming for remote education and training," in *Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6101–6106, Seattle, WA, USA, May 2015.

[9] S. Rajapaksha, V. Illankoon, and N. D. Halloluwa, "Responsive drone autopilot system for uncertain natural language commands," in *Proceedings of the 2019 International Conference on Advancements in Computing (ICAC)*, pp. 232–237, Malabe, Sri Lanka, December 2019.

[10] U. U. S. Rajapaksha and C. Jayawardena, "Ontology based optimized algorithms to communicate with a service robot using a user command with unknown terms," in *Proceedings of the 2020 2nd International Conference on Advancements in Computing (ICAC)*, vol. 1, pp. 258–262, Colombo, Sri Lanka, December 2020.

[11] H. L. Waruna Bandara, D. S. Wijesekera, and H. D. Bandara Herath, "Methodology for coping with uncertain information contained in natural language instructions in a robotic system," in *Proceedings of the 2020 2nd International Conference on Advancements in Computing (ICAC)*, vol. 1, pp. 234–239, Colombo, Sri Lanka, December 2020.

[12] U. U. S. Rajapaksha, C. Jayawardena, and B. A. MacDonald, "Ros supported heterogeneous multiple robots registration and communication with user instructions," in *Proceedings of the 2022 2nd International Conference on Advanced Research in Computing (ICARC)*, pp. 102–107, February 2022.

[13] D. D. Rajapaksha, M. N. Mohamed Nuhuman, and S. D. Gunawardhana, "Web based user-friendly graphical interface to control robots with ros environment," in *Proceedings of the 2021 6th International Conference on Information Technology Research (ICITR)*, pp. 1–6, Tokyo, Japan, August 2021.

[14] U. S. Rajapaksha, C. Jayawardena, and B. A. MacDonald, "Ros based multiple service robots control and communication with high level user instruction with ontology," in *Proceedings of the 2021 10th International Conference on Information and Automation for Sustainability (ICIAfS)*, pp. 381–386, Negombo, Sri Lanka, August 2021.

[15] U. K. Rajapaksha, C. Jayawardena, and B. A. MacDonald, "Ros based heterogeneous multiple robots control using high level user instructions," in *Proceedings of the TENCON 2021-2021 IEEE Region 10 Conference (TENCON)*, pp. 163–168, Auckland, New Zealand, December 2021.

[16] A. Buscarino, L. Fortuna, M. Frasca, and A. Rizzo, "Dynamical network interactions in distributed control of robots," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 16, no. 1, Article ID 015116, 2006.

[17] I. Tiddi, E. Bastianelli, and G. Bardaro, "An ontology-based approach to improve the accessibility of ROS-based robotic systems," in *Proceedings of the Knowledge Capture Conference, K-CAP 2017*, Austin, TX, USA, December 2017.

[18] I. Tiddi, E. Bastianelli, and G. Bardaro, "A user-friendly interface to control ROS robotic platforms," *CEUR Workshop Proceedings*, vol. 2180, 2018.

[19] M. Pomarlan and J. Bateman, "Robot program construction via grounded natural language semantics & simulation robotics track," in *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, vol. 2, pp. 857–864, Stockholm, Sweden, July 2018.

[20] M. Amaratunga, G. Wickramasinghe, and M. Deepal, "An interactive programming assistance tool (IPAT) for instructors and novice programmers," in *Proceedings of the 2013 8th International Conference on Computer Science Education*, pp. 680–684, Sri Lanka, 2013.

[21] M. A. V. J. Muthugala and A. G. B. P. Jayasekara, "A review of service robots coping with uncertain information in natural language instructions," *IEEE Access*, vol. 6, pp. 12913–12928, 2018.

[22] C. J. Sutherland and B. MacDonald, "RoboLang: a simple domain specific language to script robot interactions," in *Proceedings of the 2019 16th International Conference on Ubiquitous Robots, UR 2019*, pp. 265–270, Jeju, Jeju, Korea, June 2019.

[23] C. Datta, B. A. MacDonald, C. Jayawardena, and I.-H. Kuo, "Programming behaviour of a personal service robot with application to healthcare," in *International Conference on Social Robotics*, pp. 228–237, Springer, Berlin, Germany, 2012.

[24] C. Jayawardena, K. Watanabe, and K. Izumi, "Probabilistic neural network based learning from fuzzy voice commands for controlling a robot," *Institute of Control, Robotics and Systems*, pp. 2011–2016, 2011.

[25] C. Jayawardena, K. Watanabe, and K. Izumi, "Knowledge acquisition by a sub-coach in a coach player system for controlling a robot," in *Proceedings of the 4th International Conference on Advanced Mechatronics*, pp. 601–606, Hokkaido, Japan, 2004.

[26] K. K. P. Gayashani, U. U. S. Rajapaksha, and C. Jayawardena, "Moving a robot in unknown areas without collision using robot operating system," in *Proceedings of the 2022 2nd International Conference on Advanced Research in Computing (ICARC)*, pp. 84–89, February 2022.

[27] M. Panagoda, M. Lokuliyanage, and A. Senarath, "Moving robots in unknown environments using potential field graphs," in *Proceedings of the 2022 2nd International Conference on Advanced Research in Computing (ICARC)*, pp. 96–101, February 2022.

[28] C. Jayawardena, I.-H. Kuo, E. Broadbent, and B. A. MacDonald, "Socially assistive robot healthbot: design, implementation, and field trials," *IEEE Systems Journal*, vol. 10, no. 3, pp. 1056–1067, 2016.

[29] C. Datta, C. Jayawardena, and I. H. Kuo, "Robostudio: a visual programming environment for rapid authoring and

customization of complex services on a personal service robot," in *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2352–2357, Algarve, Portugal, October 2012.

[30] J. Kim, M. Jang, and J. Sohn, "An ontological approach for natural language command interpretation and its application in robotics," in *Proceedings of the 2006 SICE-ICASE International Joint Conference*, pp. 3874–3878, Busan, Korea, October 2006.

[31] A. Scibilia, N. Pedrocchi, and L. Fortuna, "Human control model estimation in physical human-machine interaction: a survey," *Sensors*, vol. 22, no. 5, p. 1732, 2022.

[32] M. Bucolo, A. Buscarino, C. Famoso, L. Fortuna, and M. Frasca, "Control of imperfect dynamical systems," *Nonlinear Dynamics*, vol. 98, no. 4, pp. 2989–2999, 2019.

[33] A. T. Rashid, M. Frasca, A. A. Ali, A. Rizzo, and L. Fortuna, "Multi-robot localization and orientation estimation using robotic cluster matching algorithm," *Robotics and Autonomous Systems*, vol. 63, pp. 108–121, 2015.

[34] A. A. Ali, A. T. Rashid, M. Frasca, and L. Fortuna, "An algorithm for multi-robot collision-free navigation based on shortest distance," *Robotics and Autonomous Systems*, vol. 75, pp. 119–128, 2016.

[35] S. K. Rajapaksha and N. Kodagoda, "Internal structure and semantic web link structure based ontology ranking," in *Proceedings of the 2008 4th International Conference on Information and Automation for Sustainability*, pp. 86–90, December 2008.

[36] U. Rajapaksha and H. Fernando, "Ontology matching and ranking: issues and research challenges in semantic web application development," in *Proceedings of the ITRU Research Symposium*, Sri Lanka, 2009.

[37] S. Rajapaksha and C. Jayasekara, "Ontology based semantic file search assistant," in *Proceedings of the 2021 10th International Conference on Information and Automation for Sustainability (ICIAfS)*, pp. 310–315, Galle, Sri Lanka, August 2021.