2018 18th International Conference on Control, Automation and Systems (ICCAS 2018)
Oct. 17–20, 2018; YongPyong Resort, PyeongChang, GangWon, Korea

Development of an Autonomous Mobile Assistant: RoboMuse 4.0

Muhammad Suhail[1] , Subramanian K[2] , Ayush Shukla[3] , Rishabjit Singh[4] , Sanjiv Gupta[5] and Subir K Saha[6]

[1]Department of Mechanical Engineering, National Institute of Technology, Tiruchirappalli, India (muhammadsuhail441@gmail.com)
[2]Department of Instrumentation and Control Engineering, National Institute of Technology, Tiruchirappalli, India (subramanian.krish611@gmail.com)
[3] Department of Electrical Engineering, Indian Institute of Technology, Delhi (shuklaayush247@gmail.com)
[4] Department of Electrical Engineering, Indian Institute of Technology, Delhi (rishabjit@gmail.com)
[5] Department of Mechanical Engineering, Indian Institute of Technology, Delhi (sanjiv1081@gmail.com)
[6] Department of Mechanical Engineering, Indian Institute of Technology, Delhi (saha@mech.iitd.ac.in)

Abstract: Over the years, mobile robots have proven to be of significant assistance to humans in various domains including domestic applications, agriculture, manufacturing, and defense. In this paper, we have focused on a domestic application, more specifically the development of a custom designed low cost mobile robot - RoboMuse 4.0, into a highly capable mobile assistant. From following a person around carrying his or her load, to acting as a mobile guide which is capable of autonomously charging itself by docking into a charging station, we have concentrated on automating the mobile functions of an assistant. This was achieved through various subtasks including simultaneous localization and mapping, autonomous exploration, point to point navigation, and autonomous docking and charging. The performance of the robot was also evaluated through various established and customized tests, the results of which have also been included in this paper.

Keywords: Mobile Robot, Personal Assistant, Mobile Guide, Autonomous Charging, Person Follower

1. INTRODUCTION

Several commercial robotic platforms are being currently used by roboticists across the world to conduct research in associated fields and to develop a number of applications. However, the lack of indigenous mobile robot systems in India has led to the market being dominated by foreign products which prove to be expensive. This acts as a major hurdle to Indian researchers restricting their contribution in the field. To overcome this problem, RoboMuse 4.0 - an indigenous mobile robot platform was developed at the Indian Institute of Technology Delhi. This low-cost mobile robot, as shown in Fig. 1, has a custom designed chassis and is equipped with a Microsoft Kinect sensor to help visualize its environment, and quadrature wheel encoders to keep track of its motion. The sensory data are processed on a laptop which generates appropriate velocity commands based on the operational goal. These are then passed onto the drive unit. The usage of the Robot Operating System (ROS) framework[1], makes the system modular and enables easier communication. The overall architecture of the system can be found in our previous publication [2].

Currently, the usage of commercial mobile robot platforms as a personal assistant is on the rise [3]. However, these platforms prove to be expensive in comparison to RoboMuse 4.0. A similar robot, Turtlebot 2, with a laptop costs nearly Rs.1,40,000 without accounting for duties, while a congruent setup for RoboMuse 4.0 would cost less than Rs.90,000. The cost split up for the entire system can be found here [2]. Thus, the development of an indigenous assistant proves to be an attractive option due its cost effectiveness.

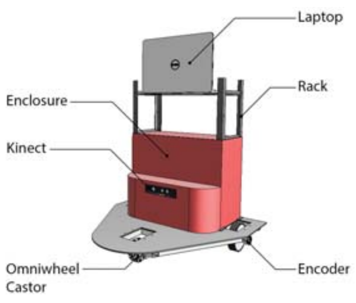


Fig. 1 RoboMuse 4.0.

The functionalities expected of a personal assistant are innumerable. It is only possible to develop such an assistant step by step, function by function. Thus in this paper, we present the development of three such functions:

- Autonomous Mobile Guidance: The robot acts as a mobile guide, autonomously guiding people to their destinations in areas/buildings that they might not be familiar with.
- Autonomous Docking and Charging: The robot autonomously detects low charge in its batteries and docks by navigating itself to a custom designed charging station, thereby charging itself.
- Person Following: The robot is enabled to follow a specific person, to whom it is calibrated to.

2. AUTONOMOUS MOBILE GUIDANCE

To act as an autonomous mobile guide, the robot should be capable of performing two major functions:

- Autonomously exploring its environment while simultaneously creating a map of it.
- Reaching its goals on the created map in the most efficient route.

These desired functionalities were developed on the RoboMuse 4.0 as three modules on a ROS framework, which have been explained in detail in this section.

2.1 Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) is the creation of a map of an unknown environment in which the robot is present, while simultaneously identifying its location within the map. This is a complicated situation as one is required for the other (a map of the environment for localizing the robot and an accurate estimate of the robot's pose for mapping). In recent times, after the advent of ranging technologies like LIDAR, stereo cameras, high resolution monocular cameras, researchers are resolving SLAM using methods involving feature extraction. They track these features between successive samples to build the map and determine the pose of the robot in it.

The Kinect sensor present on the robot furnishes point cloud (PCL) data of its field of vision to a ROS package, *pointcloud_to_laserscan*, which emulates a laser scan from the PCL data. While this laser scan helps the system map the environment, localization is achieved by fusing the PCL received from the Kinect with the odometry data from the wheel encoders. These data are supplied to the *rtabmap_ros* package which performs Real Time Appearance Based Mapping to generate a 3D PCL map, and a 2D Cost map[4], as shown in Fig. 2 and Fig. 4, respectively. A cost map is a 2D map of the environment in ROS, which is presented as a grid, hosting information about free spaces, obstacles, and unexplored spaces, as a value between 0 to 255 mapped linearly. RTAB uses an RGB-D graph based approach to add new locations to memory, after comparing feature points found by Scale Invariant Feature Transform (SIFT) descriptor, with locations in long term memory. In the event that the number of common features between the current scene and previous scenes in memory crosses a defined threshold, loop closure is flagged. Loop closure is a mechanism to correctly identify whether the robot has previously visited the location. This concept helps in adjusting localization errors and optimizes the generated map.

2.2 Autonomous Exploration

While SLAM helps to map the portion of the environment which falls within the field of vision of the robot's sensor, it is the responsibility of the exploration module to ensure that the entirety of the robot's environment is mapped, by strategically moving the robot around the environment. This exploration can be done by means of teleoperating the robot. However, a completely independent system would be able to produce a map of its environment without any human intervention.

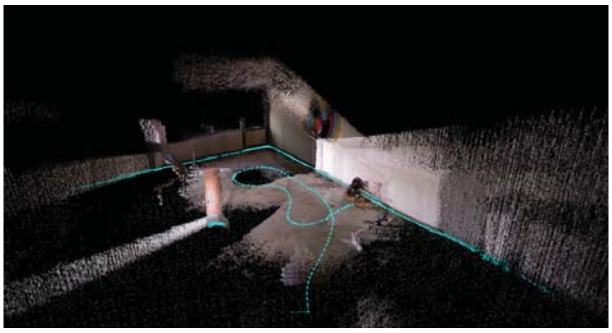


Fig. 2 3D PCL Map as produced by RoboMuse 4.0.

A frontier based algorithm for exploring neighboring zones in an optimized manner was proposed by Yamauchi B[5] and ROS provides a completely customizable package, *frontier_exploration*, to run this technique. It is necessary to execute this package on top of a fully configured navigation stack. Navigation stack is a collection of algorithms which takes in the odometry data, sensor streams, goal location, and outputs the velocity to the mobile base. The *explore_server* node accesses the costmap (Fig. 3) and performs a Breadth First Search for frontiers and returns the list of frontiers to the *explore_client* node. The *explore_client* node provides the closest frontier as an action goal to the *move_base* package which plans a secure path to reach it. *move base* is a package associated with the navigation stack and its functioning is explained in detail in the next subsection.

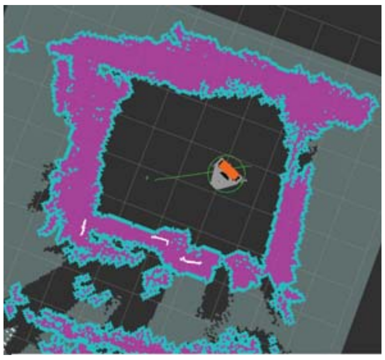


Fig. 3 Costmap from Frontier Exploration.

2.3 Point to Point Navigation

The SLAM module, in combination with the autonomous exploration module, has resulted in the creation of a map of the robot's environment within which the robot is accurately localized. Thus, the point to point navigation module is responsible for ensuring that the robot safely reaches its goal/destination from its current pose avoiding all static and dynamic obstacles in its path.

Positions of significant landmarks that the robot would visit during guidance are marked from the map onto the back end of the user interface. Upon acquiring target location from the user as input, the goal pose is passed onto the *move_base* package.

The *move_base* package encompasses a global and a local planner to implement path planning between the current pose and the goal pose. The local planner assembles a local costmap with the current sensor data which is used for short term controller decisions that are responsible for steering around momentary obstructions. The global planner exploits stored map information to piece together a complete global plan by means of conventional path planning algorithms. RoboMuse 4.0 employs a global planner based on Dijkstra's algorithm and a local planner based on the Trajectory Rollout approach[6]. The plan created by *move_base* package is converted to motor velocities by a node in the package.

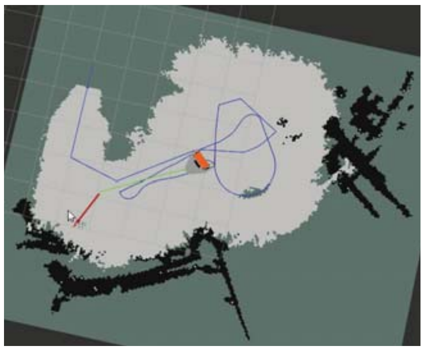


Fig. 4 Costmap of the environment - Path traversed during exploration (blue), goal (red), path planned to reach goal (green).

3. AUTONOMOUS DOCKING/CHARGING

Thoroughly automated systems possess the capability to recharge their power source upon identification of decline in battery voltage below safe limits of operation. RoboMuse 4.0 can charge its lead-acid battery by effecting electrical contact with the custom made charging dock (Fig. 5).

3.1 Dock Hardware

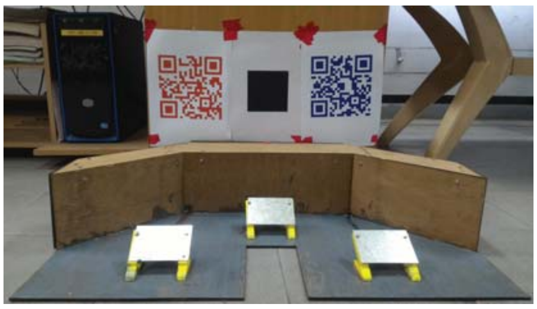


Fig. 5 Custom Charging Dock.

The idea is to have metal pedals on the charging dock which would mate with the terminals attached to the robot when the robot is in charging position. The robot's terminals are connected to the battery via a relay which is switched through the microcontroller. Further, a flexure mechanism was used to provide a spring action so that a constant normal force exists to ensure contact between the pedals and the terminals.

Conventional lead-acid battery chargers apply a constant voltage to the battery and charge the battery at around 1A. Since we wanted RoboMuse 4.0 to run continuously, we made sure that the charging time was not very long. Hence, we adopted a three-step fast-charging method for charging the robot. For this purpose, we used a dedicated lead acid charging IC, BQ2031 from Texas Instruments. The charger operates in three stages:

1. Constant Current Stage
2. Constant Voltage Stage
3. Maintenance Stage

By tweaking the different resistor combinations in the circuit (Fig. 6), we arrived at a charging current of 2.5A in the constant current stage. The circuit automatically detected the presence of a battery and subsequently supplied voltage. It also switched to maintenance stage when the battery was charged.

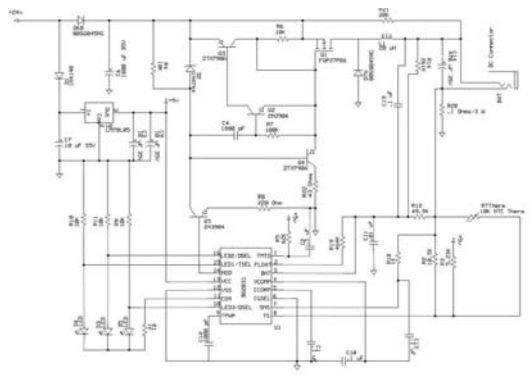


Fig. 6 Battery Charger Schematic.

3.2 Docking Algorithm

The dock is anchored to a single location in the area and has a fixed position in the costmap. Although the robot is aware of the dock's position, moving to within the allowed tolerance and autonomously establishing contact with the metal plates of the charging dock, is a challenging affair due to the probabilistic nature of localization. Therefore, we used a visual feedback approach which relies on the Kinect's image data. The procedure has been broadly summarized below.

Upon initiation of the docking procedure, as we are already aware of the dock's approximate pose within the map, a goal pose 2 metres ahead of the dock was provided to the *move_base* package. The errors associated with localization brings the robot into the vicinity of the dock. Considering the demand for high precision, a two phase docking algorithm was devised.

3.2.1 **Phase one: Orienting with minimal error**

Currently, we are approximately 2 metres in front of the dock. The major setback at this stage is that, due to the uncertainties associated with localization, the robot might be incorrectly aligned with the axis of the dock or might be laterally offset from the axis. Thus, this phase of docking involves guiding the robot to align itself in the direction of the dock with minimal lateral offset, as shown in Fig. 7. To carry out this part, we have employed QR codes as they offer substantial feature set for marker detection in an image.

Two distinctly colored QR codes were attached slightly above the dock on either sides at equal distances from the dock's center. The differently colored QR codes helped distinguish the left and right sides of the dock. The ROS package *find_object_2d* implements feature descriptors and detectors to identify the pose of the markers with reference to the Kinect[7].

Algorithm:

The points $p_1(x_1,\ y_1,\ z_1)$ and $p_2(x_2,\ y_2,\ z_2)$ are the centers of the left and right QR codes respectively. Then, the midpoint of p_1 and p_2 in the ground plane, p_{mid}, would be the position of the dock. Thus we define the final goal position (x,y) for the robot at a point 1.5m ahead of p_{mid} (x_{mid},y_{mid}) and in a direction perpendicular to the line joining p_1 and p_2. From analytical geometry,

$$x = x_{mid} - \sqrt[2]{\frac{1.5^2}{(\frac{x_1-x_2}{y_2-y_1})^2+1}} \tag{1}$$

$$y = y_{mid} + (\tfrac{x_1-x_2}{y_2-y_1})(x - x_{mid}) \tag{2}$$

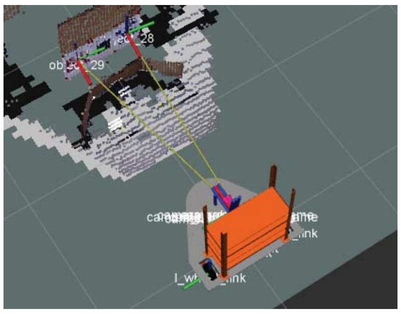


Fig. 7 Robot pose at the end of Phase I.

The calculated values of the final goal pose are provided to the *move_base* package. Small lateral error in real time arises due to minor errors in localization. This was corrected in phase two with the help of visual feedback.

3.2.2 **Phase two: Docking with closed loop feedback**

In this phase, the robot takes advantage of the Kinect's visual data by using it as feedback.

Algorithm:

The angular velocity of the robot is controlled by a proportional controller, which signals the base to turn in proportion to the difference between the distance of the left and right QR codes from the robot (error signal).
d_1: The distance between the robot and p_1.
d_2: The distance between the robot and p_2.
d_{mid}: The distance between the robot and p_{mid}.

$$v_{linear} = constant \quad \forall \quad d_{mid} > Threshold \tag{3}$$
$$v_{angular} = K_p * (d_1 - d_2) \tag{4}$$

As the robot starts moving slowly towards the dock, its heading is corrected repeatedly. Over a period of time, it converges onto the desired heading which is perpendicular to the line joining the two QR codes. Once the d_{mid} value lowers below the threshold value, the robot stops as it has docked successfully.

4. PERSON FOLLOWER

Robotic systems nowadays are expected to function alongside humans instead of replacing them. The person following function allows the robot to follow the user at a constant distance of 1.5 metres, varying its speed based on how fast the person walks. This is a useful feature which allows the robot to carry people's loads for them and follow them around.

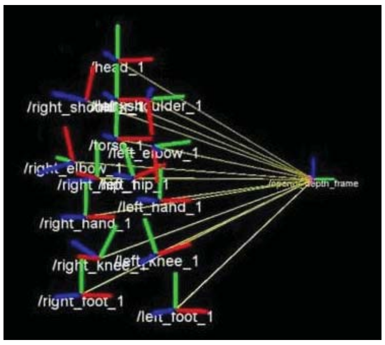


Fig. 8 Frames returned by NITE[8].

NITE, a motion tracking middleware developed by PrimeSense, performs scene analysis (separation of users from background), ground PCL elimination, filtering, and classification on Kinect's visual data, to return frames of several parts of the body like torso, head, ankles etc.(Fig. 8). The frame of the torso was looked up with respect to base frame, i.e., position of the person with respect to the robot was derived. Upon receiving user's pose, two independent PID loops, one controlling linear velocity and the other controlling angular velocity, are run in the background.

The error for the linear velocity controller is the distance between the robot and the person, while the error for the angular velocity controller is the angle between the robot's heading/direction and the line joining the person's torso and the centre of the robot. On the occasion that the person to be tracked moves out of visibility of the Kinect, *openni_tracker* returns last known pose repeatedly. In such a case, the robot stops abruptly to prevent accidents.

5. PERFORMANCE ANALYSIS

The international standards for mobile robots are under development and existing standards grant only a qualitative and abstract idea with reference to safety measures (ISO 13482)[9]. We have devised a set of simple experiments with a view to quantitatively analyze the accuracy and repeatability of RoboMuse 4.0 corresponding to its applications. The scheme and results of these tests are presented in this section.

5.1 Odometry Optimization

An ideal scenario is where the estimated odometry data as given by the sensors mounted on the robot and the ground truth position of the robot are coincidental. However, due to slippage of the wheels, design errors in optical encoders, unequal radii of wheels, and other errors, odometry published deviates from the actual position of the robot. Borenstein et al. suggested a standardized University of Michigan Benchmark(UMBmark) test[10] to identify the systematic errors and a method to correct them.

According to the UMBmark test, the robot is to be run on a square pattern of side length 4m, in both clockwise and anticlockwise directions. The error at the conclusion of the run between the estimated position and the absolute position of the robot (ground truth) in the real world is recorded for multiple trials in both directions of movement. The radii of the wheels and distance between the wheels is updated in the formulae used for calculating odometry, by applying a correction factor calculated from the error data as suggested in their paper. On repeating the test after applying the correction factor, we noticed that the error in odometry was minimized substantially. The UMBmark average error results before and after application of correction factors are shown in Table 1.

Table 1 UMBmark Average Error Results.

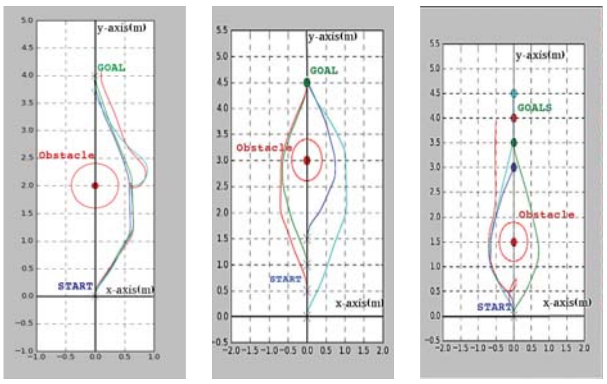
Direction	Pre-Correction		Post-Correction	
	X(cms)	Y(cms)	X(cms)	Y(cms)
Clockwise	-16.216	-25.492	-2.942	4.452
Anti-Clockwise	-26.05	42.93	-0.95	2.396

5.2 Mobile guidance navigational parameters

To determine the quantitative parameters of the mobile guidance function of the robot we conducted two major tests:

5.2.1 Distance of closest approach to obstacles

As the robot has to safely traverse the environment, it is imperative that the robot maintains a minimum distance from any obstacles in its path. To study the distance of closest approach to obstacles in the robot's path, we implemented the point to point navigation module for 3 different cases with multiple trials in each case. The obstacle pose and the path taken by the robot was recorded for each trial as seen in Fig. 9. The test cases were:



(a) Case I. (b) Case II. (c) Case III.

Fig. 9 Paths taken during cases under study.

- Case I: Fixed start pose and fixed goal pose.
- Case II: Varying start poses with fixed goal pose.
- Case III: Fixed start pose with varying goal poses.

The average distance of closest approach calculated for each case has been tabulated in Table 2.

Table 2 Average closest approach distance to obstacles.

Case	Minimum Approach Distance (m)
I	0.3841
II	0.5434
III	0.4383

As seen from the table, the robot maintains considerable distance from any obstacle in its environment, ensuring safe operation.

5.2.2 Accuracy and repeatability of point to point navigation:

To determine the accuracy and repeatability, the robot was programmed to move to different goal locations in the presence and absence of obstacles in its path. Assuming the robot's initial position was (0,0), multiple trials of the following test cases were done and their results have been tabulated in Table 3.

- Case A: Goal at (4m, 4m) with no obstacle in the path.
- Case B: Goal at (4m, 4m) with a static obstacle at (2m, 2m).
- Case C: Goal at (4m, 4m) with a dynamic obstacle in the path.

Error was calculated as the difference between the goal/target position and the final position reached by the robot after executing the planned motion. In addition, repeatability was defined as the standard deviation of the errors, while accuracy was defined as the absolute value of the mean error.

Table 3 Accuracy and Repeatability.

Case	Absolute Mean Error		Standard Deviation of Error	
	X(cm)	Y(cm)	X(cm)	Y(cm)
A	14.08	19.3	7.364	6.605
B	19.26	12.52	2.225	2.307
C	17.84	15.2	8.57	4.101

The robot gives good results in terms of accuracy and repeatability, and the errors associated are minimal. As a result, the mobile guidance function remains unaffected.

5.3 Docking accuracy

Both the phases of the docking procedure were tested for their accuracy and repeatability. While success rate is the equivalent of accuracy for our case, repeatability for pass/fail experiments is not clearly defined. Hence, we followed the principle of accordance as introduced by SD Langton[11]. Accordance is the equivalent of repeatability and it is defined as the following ratio:

$$A = \frac{x(x-1)+(n-x)(n-x-1)}{(n)(n-1)} \quad (5)$$

where *x : Number of successes, n : Number of Trials, A : Accordance*.

For Phase I testing, we defined success as the ability of the robot to reach the 1.5m mark with a lateral error of less than 20cm.

Table 4 Phase I Performance.

Offset Tolerance(cm)	Success Rate	Std. Devn.	Accordance
15	9/10	0.094	0.8

In Table 4, Offset Tolerance refers to the maximum lateral offset of the robot from the axis of the charging dock before the commencement of phase I. It was observed that as lateral offset increased beyond 15cm, there was a significant reduction in docking accuracy.

For Phase II testing, we defined success as the ability of the robot to dock accurately with perfect mating of the robot's terminals and the dock's pedals.

Table 5 Phase II Performace.

Approach Distance(m)	Success Rate	Std. Devn.	Accordance
1.0	8/10	0.126	0.664
1.5	10/10	0.00	1.00
2.0	8/10	0.126	0.664

In Table 5, Approach Distance refers to the distance of the robot from the dock, along the dock's axis before the commencement of Phase II. After experimenting with multiple approach distances, 1.5m was found to be ideal due to maximum accordance.

From the tables, we understand that the robot has high docking accuracy as long as the lateral offset at the beginning of Phase I is within the tolerance limit of 15cm.

5.4 Person Follower Parameters

As it is not possible to define accuracy and repeatability for this function, the corner case parameters have been specified. On experimentation, it was found that tracking is lost at a minimum separation distance of 87.5cm and a maximum distance of 4.87m from the robot.

This function is extremely robust and the measured parameters indicate that the robot will be able to track a person even for long distances of separation.

6. CONCLUSIONS

RoboMuse 4.0 was developed as a low cost indigenous mobile platform that could be used for the development of a variety of applications and testing algorithms. In this paper, we have demonstrated the development of the robot into a mobile assistant. The developed functionalities find their use in a variety of scenarios, from giving people a guided tour of a museum or art gallery to helping one carry his or her load. Several tests were conducted and their results were studied to ensure that the system was robust and functioning upto its maximum capability. A personal assistant must posses diverse abilities, thus, we shall be focusing on developing more such relevant features on RoboMuse 4.0 in near future. We envisage that RoboMuse 4.0 shall combine and coordinate with other similar robots to complete complicated and time consuming tasks.

REFERENCES

[1] Quigley, Morgan, et al. "ROS: an open-source Robot Operating System," *ICRA workshop on open source software*, Vol. 3. No. 3.2. 2009.

[2] A. Shukla, R. Singh et al. "Development of a Low-Cost Education Platform: RoboMuse 4.0," *Advances In Robotics*, 2017.

[3] PR2. (n.d.). Retrieved June 11, 2018, from *http://www.willowgarage.com/pages/pr2/overview*

[4] M. Labbe and F. Michaud, "Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation,"*IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 734-745, 2013.

[5] Brian Yamauchi, "A Frontier-Based Approach for Autonomous Exploration", *Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Monterey, CA, July 1997, pp. 146-151.

[6] Trajectory Rollout Approach. (n.d.). Retrieved June 11, 2018, from *https://wiki.ros.org/base_local_planner*

[7] Find-Object. Retrieved April 1, 2016, from *https://introlab.github.io/find-object/*

[8] Screenshot. (n.d.). Retrieved June 11, 2018, from *https://www.youtube.com/watch?v=9VcihcrvLRI*

[9] ISO13482. (n.d.). Retrieved June 11, 2018, from *https://www.iso.org/standard/53820.html*

[10] Borenstein, J. and Feng, L., "Measurement and Correction of Systematic Odometry Errors in Mobile Robots," *Proc. of. IEEE Transactions on Robotics and Automation*, vol. 12, no. 6, pp. 869-890, 1996.

[11] Langton, S. D et al., "Analysing collaborative trials for qualitative microbiological methods: accordance and concordance,"*International Journal of Food Microbiology* 79, pp. 175-181, 2002.