Autonomous Robot Based on Robot Operating System (ROS) for Mapping and Navigation

Mohan Kumar N, Shreekanth T
Department of Electronics and Communication Engineering
Shri Jayachamarajendra College of Engineering
Mysuru, India
mohan.24manu@gmail.com, speak2shree@gmail.com

Sandeep B
Robert Bosch Engineering and Business Solutions Private Limited
Bengaluru, India
bsandeepmit@gmail.com

Abstract— Mobile robotics is a research area that has witnessed incredible advances for the last decades. Within this well-studied domain of mobile robotics, our area of interest is robots that inhabit in unmodified office-like environments that are designed for and shared with people. Issues like mapping, self-localization, and autonomous navigation in an indoor environment are popular issues in the field of autonomous robots and have been deeply studied for the past few years. The presented work describes how a Robot Operating System ROS-based control system is used with a mobile robot for self-localization, indoor mapping and autonomous navigation in an office like environment.

Keywords – Mobile, Robotics, Mapping, Localization, Navigation, ROS.

I. INTRODUCTION

Mobile robots can be "autonomous" which means they are capable of navigating in an uncontrolled environment. An autonomous robot [10] is a robot that performs its tasks with high degree of autonomy. A fully autonomous robot will have the ability to obtain information about its environment, work for an extended period without human intervention and to avoid situations that are harmful to its environment or to itself. An autonomous navigation system is an on-board, integrated suite of sensors and technology that enables perception, path planning and autonomous navigation capabilities [1].

Although we have seen many robots with autonomous navigation feature after been provided with a static map created priory using SLAM approach, we haven't witnessed many with the feature of mapping the environment autonomously. To develop such a mobile robot systemwhich does localization, mapping, path planning and navigation effectively and efficiently in an office like environment, the challenge however in office like environments is that most of the area (cubicles in the office floor) looks identical in nature. So we have to create a map which would then help the robot to locate itself in the map and navigate autonomously.

Hence a need is aroused to develop a robot which could map its environment autonomously avoiding all possible obstacles in its path. In this proposed work, a mobile robot is used as a test bench to understand various components and aspects of a mobile robot [12]. The mobile robot configured as autonomous robot is a differential drive non-holonomic robot. It consists of various sensors onboard such as the Yaw rate sensor, Wheel Encoder for odometry etc.

Thus in the proposed work an autonomous guided navigation system implemented on a mobile robot chassis with Robot Operating System (ROS) software counterpart intended for indoor navigation is presented and is designed to work both in autonomous as well as in manually controlled modes.

The Robot Operating System (ROS) is an open source framework for robotics software development with its roots at Willow Garage and Stanford University [13]. The philosophy is to make a piece of software that could work in other robots by making little changes in the code without much effort so that we do not have to reinvent the wheel. It consists of modular tools divided into libraries and supports different languages such as C++, Python and LISP. The version of ROS we are working on, at the time of writing this thesis is ROS *Hydro Medusa*.

The sensors and actuators used in robotics have also been adapted to be used with ROS. ROS provides standard operating system facilities such as hardware abstraction, low-level device control, implementation of commonly used functionalities, message passing between processes, and package management. It is based on graph architecture with a centralized topology where processing takes place in nodes that may receive or post data in the form of standard message formats, such as multiplex sensor, control , state, planning, actuator, and so on.

Many of the capabilities frequently associated with ROS are the libraries which gives a powerful set of tools to work with ROS easily. Of these, navigation library, gmapping library, rviz visualizer, simulators, and debugging tools are the most important ones [11].

Thus a mobile robot system based on ROS is developed to generate a 2D map of the office floor and use the same map to navigate from one point to another point. Firstly, the mobile robot is made to move around office floor in order to map, avoiding obstacles. This map is further stored as a reference data with proper distinction between high cost area and low cost area; which is used later by the mobile robot to navigate within the mapped office floor area. Listed below are the approaches followed to achieve the objectives stated above.

- Manual Mapping using ROS's Gmapping Stack
- Autonomous Mapping with State Machine Implementation using ROS's Gmapping Stack.
- Navigation using ROS's Navigation Stack.

The mapping, localization and navigation functionalities are achieved by utilizing a kinect sensor, ultrasound sensor, odometry and yaw rate sensor data. The system is implemented in a loosely coupled set of ROS packages, allowing for the swapping out of hardware and software with minimal effort. A robust state machine model is developed to support the autonomous mapping feature.

The following sections explain how a ROS based autonomous robot is developed and the objectives are accomplished.

II. RELATED WORK

A. Localization and Mapping

Robot localization denotes the robot's ability to establish its own position and orientation within the frame of reference. This position may be relative to features in the environment, such as the starting position, goal, or other arbitrary point. A thorough discussion of the reasons for localization is provided by Negenborn in his Master's Thesis [2].

For autonomous navigation the robot should construct a map or floor plan and localize itself in it. Camera, ultrasonic sensors or laser can also be used for mapping. SLAM (Simultaneous Localization and Mapping) [3] [4] is one of the popular localization and mapping algorithm widely used in the world of robotics. SLAM is the process by which a mobile robot can build a map of an environment and at the same time use this map to compute its own location. SLAM's ability to provide mapping and localization simultaneously has made it a technique of tremendous value in the field of robotics [5].

B. Navigation and Obstacle Avoidance

Success in navigation requires success at the four building blocks of navigation: perception, localization, cognition and motion control. With all the perceived information from sensors, the robot has to construct a two-dimensional geometric representation (grid map) of its environment using the laser scanner. It utilizes a combination of this geometric data and odometry information supplied through the wheel encoders to determine its current location [6].

Once there is map in place the mobile robot tries to navigate through the map area keeping map as a reference. While navigating the robot tries to follow global plan to plot a path to those desired coordinates. There are various global planner for navigating a robot across a map area to name a few [9] Dijkstra's algorithm, A* algorithm etc. Once a global plan has been generated, the local planner translates this path into velocity commands for the robot's motors. It does this by creating a value function around the robot, sampling and simulating trajectories within this space, scoring each simulated trajectory based on its expected outcome, sending the highest-scoring trajectory as a velocity command to the robot, and repeating until the goal has been reached avoiding all the obstacle in the way [7].

A similar paper work [8] was published by Willow Garage in 2010 where a mobile robot used ROS to autonomously navigate more than 26 miles in an office environment.

III. METHODOLOGY OF THE PROPOSED SYSTEM

The objective of the proposed work is to develop a robot that uses ROS (Robotic Operating System) framework that could autonomously map and navigate through the area of interest. With this objective in mind, a mobile robot system is proposed as shown below. The Fig 1, shows the conceptual block diagram of ROS based autonomous robot.

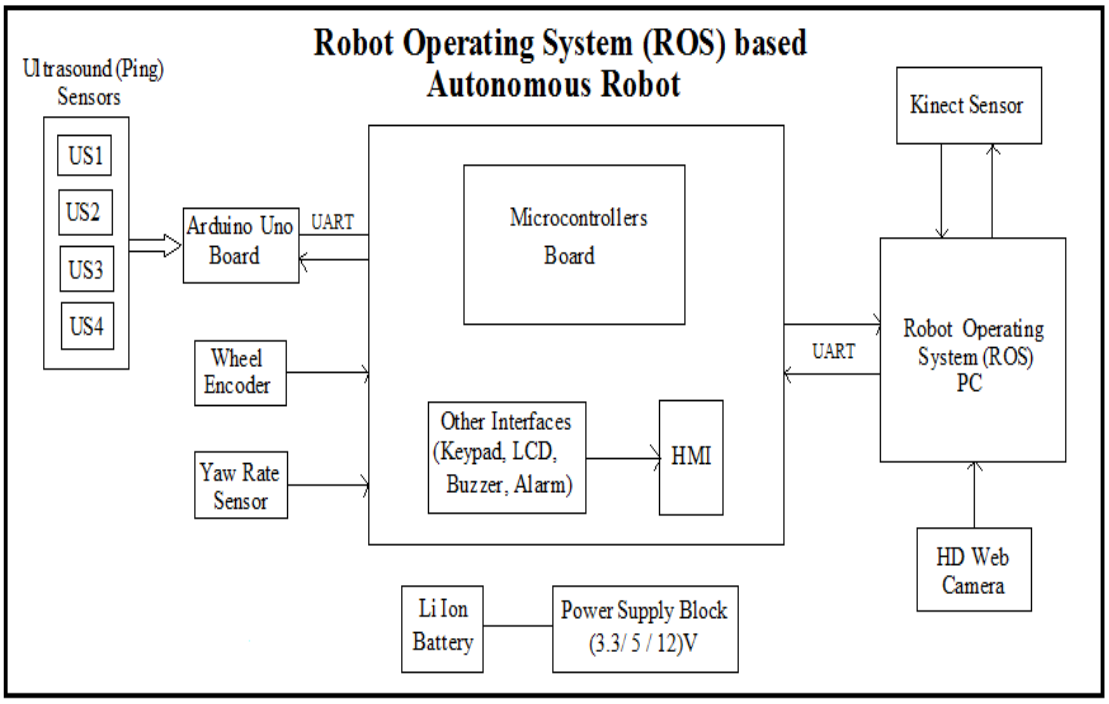

Figure 1. Block diagram representation of Autonomous Robot

A. Design Process of ROS based Autonomous Robot

Following section lists down some of the changes made across mechanical, electronics and software counterparts.

1) Mechanical

- Four pillars of aluminium bar is taken from bottom of the chassis to provide a platform to support laptop to host ROS and to mount Kinect sensor.
- A webcam is mounted on one of the side bars.

2) Electronics

- Interfacing four ultrasonic sensors using Arduino Uno microcontroller board.
- Proper wire harnessing to obtain ultrasonic sensor data.
- A 12V regulator to provide constant voltage to kinect sensor.

3) Software

- Mapping, localization and autonomous navigation are accomplished using ROS provided packages.
- A state machine implementation for autonomous mapping and navigation
- A driver code to obtain all the sensor data over UART line.
- A URDF file to provide physical description of the autonomous robot.
- Node to extract cubicle id and other text, populate markers on the map indicating possible openings and functionality to distinguish between explored and unexplored area.

Fig 2 shows the image of the complete assembly of the mobile robot.

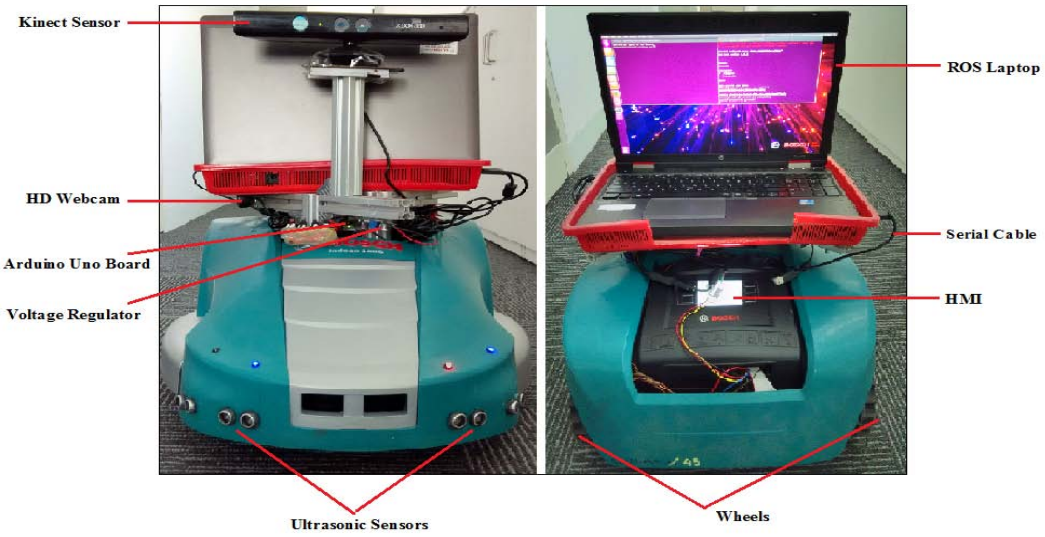

Figure 2: Images of mobile robot with mechanical changes showing various components

IV. IMPLEMENTATION

A. Interfacing of Mobile Robot with ROS PC

An API to interface mobile robot with the ROS PC which helps in proper interaction between the hardware (mobile robot) and software (ROS framework) is developed. In this module the data acquired from the sensors are combined and sent it across to ROS PC through UART lines for further utilization for various tasks. In the same module, control commands sent from ROS PC is received and interpreted accordingly. This module sends and receives data on UART lines with proper error checking mechanism (CRC).

B. ROS Code Structure Features and Potential

The ROS packages presented offers several features, many of which are inherited by the direct integration with ROS middleware [9] [13]. These packages enable the interface with ROS tools for data processing and analysis of the mobile robot platform, like 3D visualization (*rviz*), logging real-time robot experiments and playing them offline (*rosbag*), plotting data (*rqt_plot*) and visualize the entire ROS network structure (*rqt_graph*).

In order to accomplish these functionalities, the current work utilizes readily available ROS standard *gmapping* and *navigation* stacks which are tested and built for efficient and effective mapping and navigation of mobile robots of various dimensions; which comes with the installation. There are few other custom built ROS packages which are created using *roscpp* library.

Listed below are the major functionalities of this proposed work carried out using the features and potentialities of ROS.

1) Manual Mapping using ROS's Gmapping Stack

In manual mapping process the robot is controlled manually through keyboard using the keys as shown in the below Fig 3 to map the area of interest.

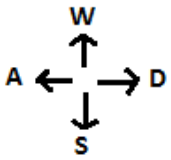

Figure 3. Commands to Control Mobile Robot.

The robot constructs a two-dimensional geometric representation of its environment using the laser scanner. It utilizes a combination of this geometric data and odometry information supplied through the wheel encoders to determine its current location. This process is achieved using SLAM approach which is part of the gmapping stack in ROS. The final output of this process is a 2D map which can be visualized using RVIZ tool.

2) Navigation using ROS's Navigation Stack

Once mapping and localization are successfully done then navigation can be easily achieved. The ROS navigation stack uses plugins for local and global planning which makes ROS very useful and very simple to

navigate in undefined environments. In this proposed work the move_base package part of navigation stack has been used for implementing path planning, which accomplished autonomous navigation.

3) Autonomous Mapping with State Machine Implementation

Autonomous Mapping process is designed using state machine concept. The state machine is implemented in such a way that, the robot gathers all the sensor information and takes decision accordingly to avoid obstacles in its path. Subsequently a 2D map of the area traversed by the robot is generated using SLAM approach similar to the one discussed in the above section.

Furthermore with the basic behavior of the autonomous mapping, additional features like extraction of cubicle id's and putting them as markers on map, alignment correction, resuming from previously stopped position(x, y, Θ), indicating possible openings, distinction between explored and unexplored area, emergency stop etc., are also implemented as part of this process.

C. Additional Features Implemented

Additionally to the main objectives, several other exiting features are also implemented inside autonomous mapping module as part of the state machine.

1) *Map Resumption Feature*

During the autonomous mapping process if the battery goes down, the mobile robot needs to be lifted from its current position and has to be kept for charging at the base station. As a result of this we lose the data of the area that has been already mapped and the entire mapping process needs to be started all over again. In order to overcome this drawback, a map resumption feature is developed.

Once the battery is charged, the user needs to take the mower back to the position from where it was lifted. Then the autonomous mapping process is launched again, the program reads the old position, orientation and the marker information from the previously saved files and forcefully initializes all the sensor inputs, maps and markers with old data. The user can easily visualize this in RVIZ tool in ROS.

V. RESULTS AND DISCUSSIONS

A. Fusing Odometry and Yaw Rate Sensor Data

To get the best possible estimate of the position the sensor data has to be fused. One of the effective approaches in the sensor fusion is Extended Kalman Filter (EKF) technique which is an extendend version of Kalman-filter. In ROS this is easily achieved using the package robot_pose_ekf. In this current work robot_pose_ekf is used to combine yaw data coming from the yaw rate sensor and the calculated yaw from odometry to obtain best estimate of the yaw data. The plots (Fig 4 and Fig 5) shows the yaw from yaw rate sensor, calculated yaw from odometry and fused yaw plotted against time.

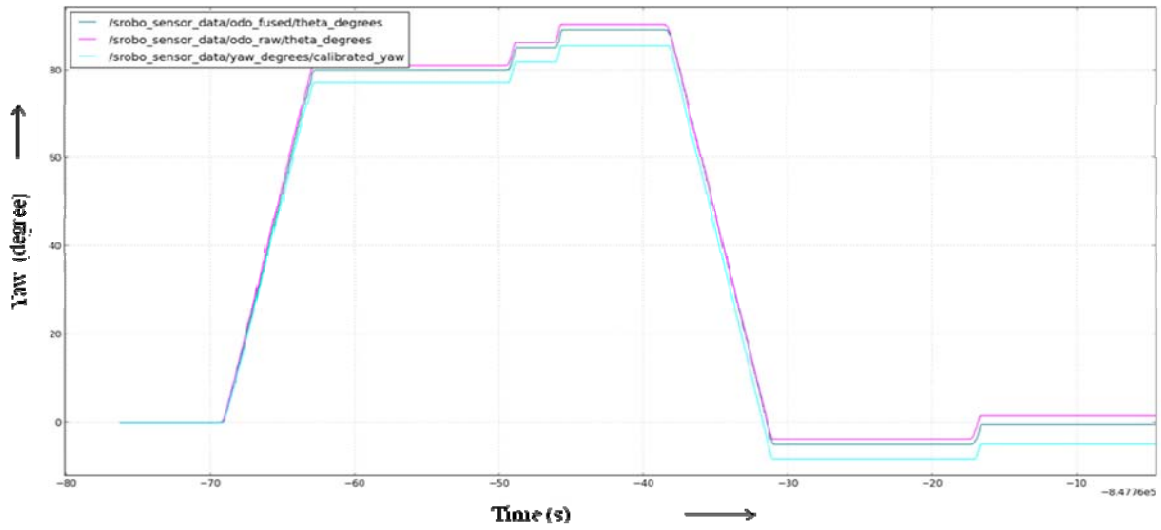

Figure 4. Plot of Yaw (degree) vs Time (s)

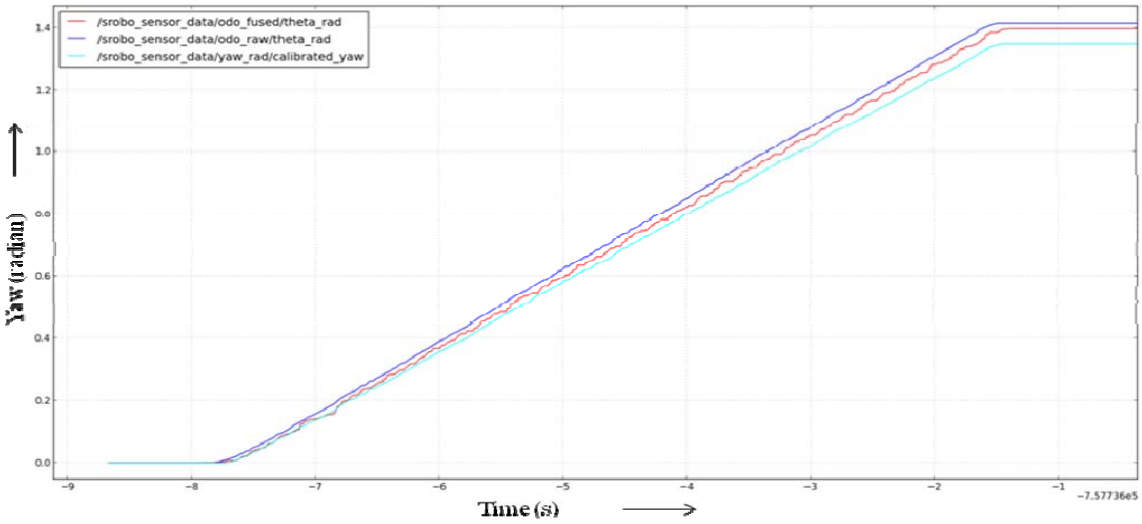

Figure 5. Plot of Yaw (radian) vs Time (s)

With this fusion, it is osbserved that the maps generated are of better quality compared to the one's using unfused data. The Fig 6 shows the comparison of maps with unfused and fused data respectively.

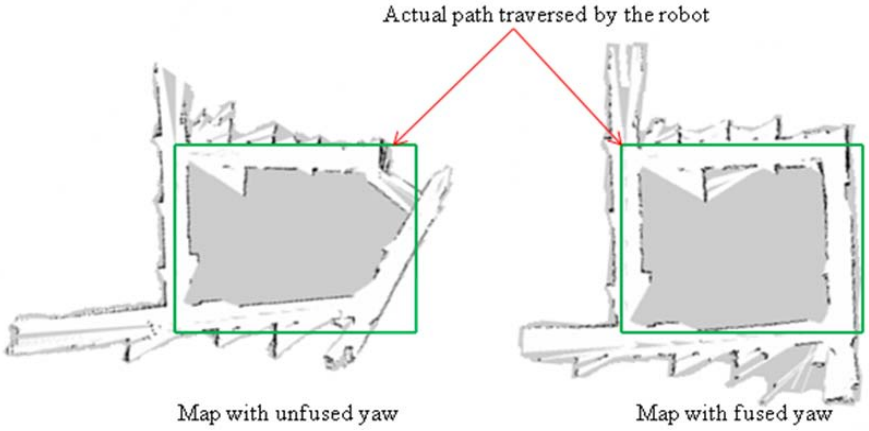

Figure 6. Comparison of fused and unfused data.

B. Manual Mapping

In this module, the 2D maps of the office floor are generated by manually guiding the mobile robot by controlling using keyboard; around the area to be mapped. Maps are created using SLAM approach with a resolution of 0.05m covering 100m x 100m area. Below are maps which are created using manual mapping process indicating its starting position and the actual path through which the robot traversed.

1) *Map – 1 :* First an attempt is made to map a bigger rectangular area without any narrow cubicle paths to get a fair idea whether there is any heading error to verify the reliability of fused odom and yaw data.

   The result of this is a map with loop closure which can be seen in the map in Fig 7.

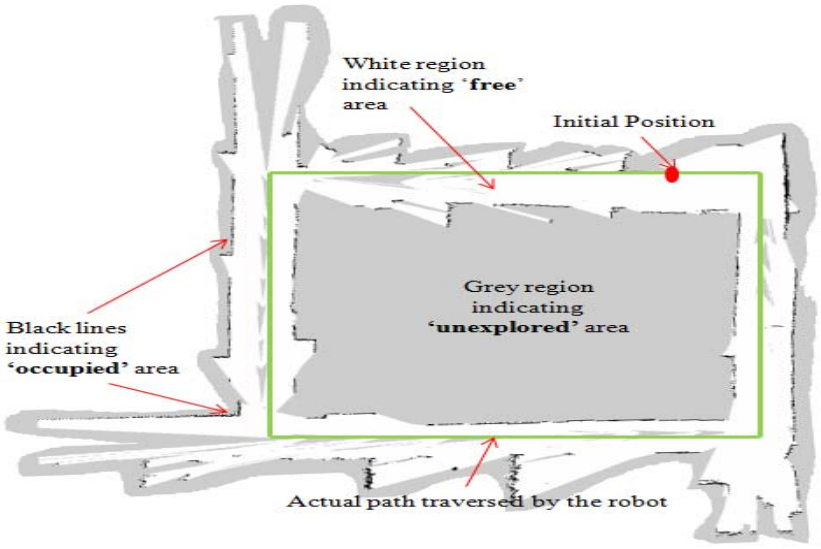

Figure 7. Manually generated Map1

2) *Map – 2:* In this case, the map is created for multiple narrow cubicle areas. The SLAM algorithm was successfully able to distinguish between narrow parallel paths (cubicle paths) and avoided overlapping of parallel paths which can be seen in the map in Fig 8.

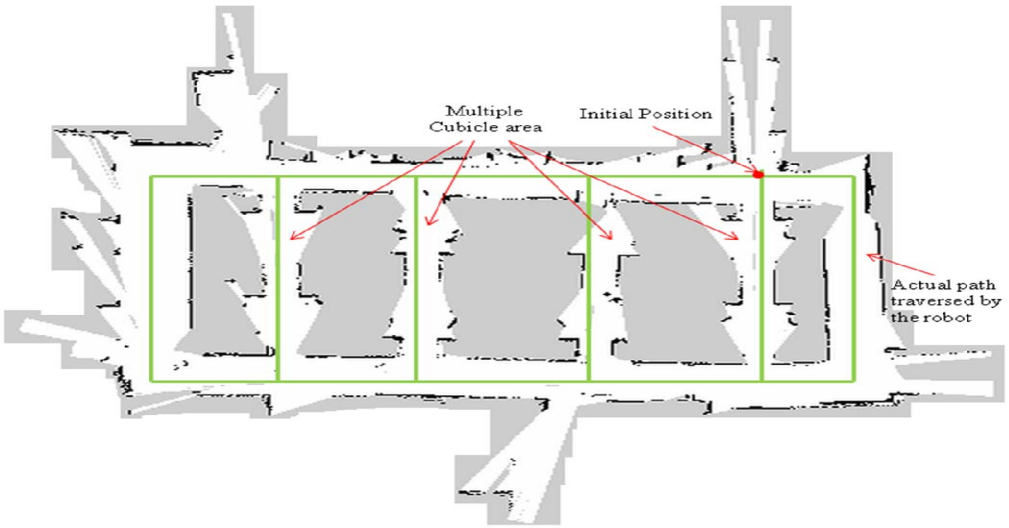

Figure 8. Manually generated Map 2

C. Autonomous Navigation

The autonomous navigation process is achieved by proper tuning of ROS's standard navigation stack for the mobile robot to emulate the robot motion, according to its kinematic model and control dimension. The screenshot below shows autonomous navigation achieved with the help of a mobile robot through an area which is mapped prior to the process. The Fig 9 shows the indication of various parameters associated with navigation process.

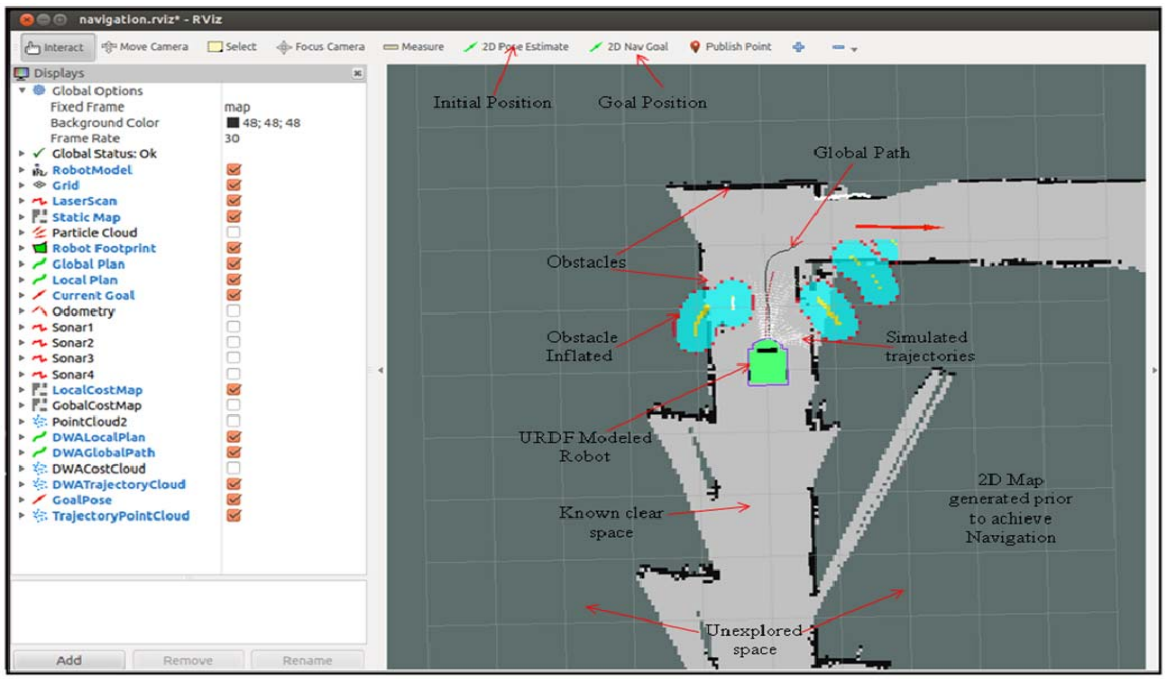

Figure 9. Screenshot of mobile robot achieving autonomous navigation in Rviz.

Given a destination and an initial pose, the navigation stack is able to plan a global path and prioritize the best possible trajectory in order to reach the destination as shown in the above screenshot.

D. Autonomous Mapping

In the autonomous mapping process maps are build autonomously combining the feature of gmapping package, which uses SLAM to build a map from sensor data published over ROS, as discussed earlier. Additionally it has the intelligence to avoid obstacle and move ahead or turn around in an area which is to be mapped. Others features like extraction of cubicle id's, indicating possible openings and putting them as markers on map are also accomplished.

1) *Map – 1:* First an attempt is made to autonomously map a simple, large rectangular area to check for heading errors and the accuracy of the map. The resulting map is shown in Fig 10.

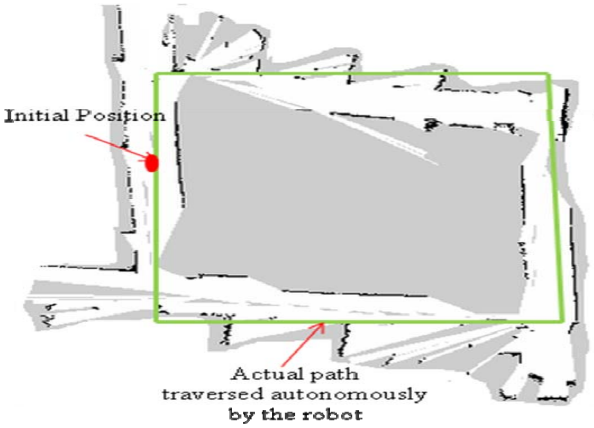

Figure 10. Autonomously generated Map 1

2) *Map – 2:* Next an attempt is made to cover a larger area along with a narrow cubicle closure. The resulting map is shown below in Fig 11. It can be observed that the heading and shape of the map resembles the actual path traversed autonomously by the mobile robot.

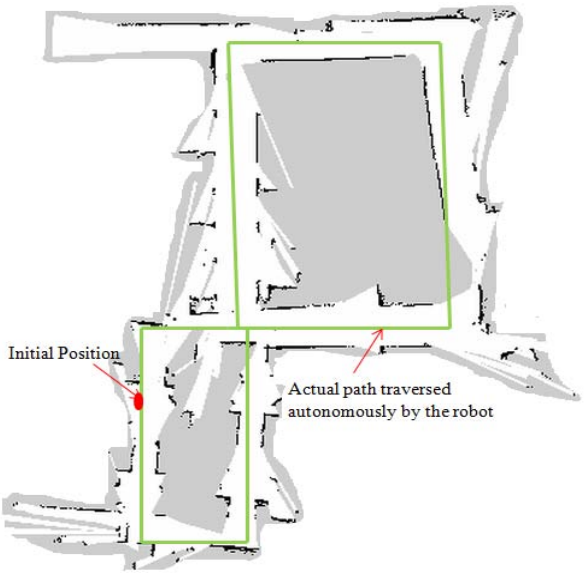

Figure 11. Autonomously generated Map 2

VI. CONCLUSION

An autonomous mapping and navigation system was designed which would autonomously map an indoor area like an office environment and later use the same map to autonomously navigate to the specified location by the user. A robust state machine is implemented to achieve the objective of autonomous mapping. Localization and mapping algorithm like SLAM have been effectively used to get reasonable and accurate maps of the desired area.

VII. ACKNOWLEDGMENT

The authors thank Mr. Nithin Ganesh, Mr. Sumanth Nirmal and Mr. Tanveer Ahmed for their valuable support, motivation and timely guidance throughout for the successful completion during the course of this proposed work.

VIII. REFERENCES

[1] Haidi Jaber, Dr.Luc Jaulin, "Robot Path Planning using Interval Analysis", ENSTA Bretagne/OSM, 2012.
[2] R. Negenborn, "Robot Localization and Kalman Filters". Master's Thesis, Utrecht University, 2003.
[3] A. Kelly. "General solution for linearized systematic error propagation in vehicle odometry in Intelligent Robots and Systems", 2001. Proceedings. 2001 IEEE/RSJ International Conference on, volume 4, pages 1938 - 1945 vol.4, 2001.
[4] Hugh Durrant-Whyte and Tim Bailey, “Simultaneous Localisation and Mapping (SLAM), Part I The Essential Algorithms”, IEEE Robotics & Automation Magazine, June 2006.
[5] Bradley Hiebert-Treuer, “An Introduction to Robot SLAM (Simultaneous Localization And Mapping)”.
[6] Dissanayake, M.W.M.G.; Newman, P.; Clark, S.; Durrant-Whyte, H. F.; & Csorba, M. (2001). “A solution to the simultaneous localization and map building (SLAM) problem.” Robotics and Automation, IEEE Transactions on, 17(3), 229-241.
[7] DereckWonnacott, MatiasKarhumaa and James Walker, “Autonomous Navigation Planning with ROS”.
[8] Marder-Eppstein, E.; Berger, E.; Foote, T.; Gerkey, B.; and Konolige, K. “The Office Marathon: Robust navigation in an indoor office environment,” Robotics and Automation (ICRA), 2010 IEEE International Conference on , vol., no., pp.300-307, 3-7 May 2010.
[9] André Gonçalves Araújo, “ROSint - Integration of a mobile robot in ROS architecture”, University of Coimbra, July 2012.
[10] S.Roland, Illah R. Nourbakhsh, Introduction to Autonomous Mobile Robots, Text Book, pp. 12-45, 2004.
[11] Aaron Martinez, Enrique Fernandez, " Learning ROS for Robotics Programming", Text Book, pp. 7-9, September, 2013.
[12] Video Tutorial by Dr. Magnus Egerstedt, Georgia Institute of Technology, “Control of Mobile Robots
[13] ROS Tutorial, ROS Documentation, Official Website: http://www.ros.org/