Chandhini K * et al.
(IJITR) INTERNATIONAL JOURNAL OF INNOVATIVE TECHNOLOGY AND RESEARCH
Volume No.5, Issue No.4, June – July 2017, 6932-6943.

Delibot with SLAM Implementation

CHANDHINI. K.
M.Tech (Computer Science and Engineering), R V College of Engineering, Bangalore, Karnataka, India

Abstract: **This paper describes and discusses a research work on "DeliBOT – A Mobile Robot with Implementation of SLAM utilizing Computer Vision/Machine Learning Techniques". The principle objective is to study about the utilization of Kinect in mobile robotics and use it to assemble an integrated system framework equipped for building a map of environment, and localizing mobile robot with respect to the map using visual cues. There were four principle work stages. The initial step was studying and testing solutions for mapping and navigation with a RGB-D sensor, the Kinect. The accompanying stage was implementing a system framework equipped for identifying and localizing objects from the point cloud given by the Kinect, permitting the execution of further errands on the system framework, i.e. considering the computational load. The third step was identifying the landmarks and the improvement they can present in the framework. At last, the joining of the previous modules was led and experimental evaluation and validation of the integrated system. The demand of substitution of human by a robot is winding up noticeably more probable eager these days because of the likelihood of less mistakes that the robot apparently makes. Amid the previous couple of years, the technology turn out to be more accurate and legitimate outcomes with less errors, and researches started to consolidate more sensors. By utilizing accessible sensors, robot will perceive and identify environment it is in and makes map. Additionally, robot will have element of itself locating inside environment. Robot fundamental operations are identification of objects and localization for conduction of the services. Robot conduct appropriate path planning and avoidance of object by setting a target or determining goal [1]. Because of the outstanding research and robotics applications in almost every segments of life of human's, from space surveillance to health-care, solution is created for autonomous mobile robots direct tasks excluding human intervention in indoor environment [2], a few applications like cleaning facilities and transportation fields. Robot navigation in environment that is safe that performs profoundly, require environment map. Since in the greater part of applications in real-life map is not given, exploration algorithm is used.**

I. INTRODUCTION

Mapping versatile mobile robots is generally characterized by representation of map and basic estimation technique. Representation of map is done using occupancy grid [9]. Grid methodologies are costly, commonly require an immense memory, for representing arbitrary objects. Representations based on feature appeal as result of compactness. In any case, it depends on highlight extractors, expect few structures in environment conditions known ahead of time. Estimation calculations are generally classified by fundamental principle. Famous methodologies include extended Kalman channels (EKFs), max likelihood technique; sparse extended information filters (SEIFs), and Rao-Blackwellized particle filters. Viability of EKF originates with way of gauging correlated posterior with point of interest landmark map, robot postures [10, 11]. Shortcoming is solid presumptions made on robot motion mode, sensor commotion. Also, landmarks are extraordinarily identifiable. Strategies [12] for managing data association information that are unknown of SLAM are disregarded, EKF filter will probably going to diverge [13]. Comparable perceptions accounted for by Julier [14] and Uhlmann. Kalman filter portrayed is for better managing non-linearities motion model of vehicle. Well-known max likelihood algorithm calculates likelihood map with historical backdrop sensor building networks system which represent to spatial limitations between robots poses [15, 16,17].

Gutmann [18] proposed a viable path of building network and recognizing loops, running max likelihood algorithm. At the point when loop is recognized, global optimization network system is done.As of late, Hahnel [19], proposed approach that track a few hypotheses of map utilizing tree association. Vital developments of tree keep approach doable for real operation. Thrun [20] proposed technique of revising robots pose in covariance inversion framework. Sparse extended information filters (SEIFs) use approximate sparsity data matrix and thus make prediction and update with consistent time. A technique presented by Eustice in delayed-state structure use precisely sparse information matrices. Paskin [21] displayed method for SLAM issue utilizing dainty trees. Along these lines, can decrease the complexity contrasted with EKF as diminished trees give linear time sifting operation. Folkessen [22] gave a powerful managing symmetries and invariants found with point of interest landmark representation. Accomplished representing every component or in metric space. Measurement subspace catches invariant of point of interest

landmark; metric space speaks to feature of data. Mapping among measurement subspace, metric space is progressively assessed and new observations are procured. Mapping considers spatial constraints among various components. This enables to regard relations to update of map estimate.

Dellaert[23] presented strategy knows as square root smoothing, mapping. Have few points of interest contrasted as EKF as have nonlinearities and compute fast. As opposed of SEIFs, gives a precisely information matrix with sparse factorization. Murphy and partners [24, 25], gave RBPF which acquainted to take care of SLAM issue. Every particle of RBPF represent to conceivable trajectory of robot. System have consequently stretched out by Montemerlo [26, 27] moving toward SLAM issue with maps. For learning Precise grid maps, RBPFs utilized by Eliazar and Parr and Hahnel [28]. Though principal work depicts representation of map, then introduces enhanced motion model which decreases particles number. In light of Hahnel, Howard gave a way for deal with grid maps of various robots [28]. Concentration is in how to consolidate data gotten with robots. Bosse [29] portray general system framework of SLAM with expansive scale conditions. Utilize a graph of local maps using co-ordinate outlines and uncertainty wrt. local frame is represented always. Along these lines, they diminish SLAM complexity problem. In this specific situation, Modayil [30] exhibited method that consolidates metrical and topological SLAM. Topology take care of closing loop issue, while metric data for neighborhood structures develop. Comparable thoughts acknowledged by Lisien et al. [31], present progressive guide with regards to SLAM.

The SLAM used in the thesis is improved one given by Hahnel [32]. Rather than utilizing settled distribution, it will compute enhanced proposal distribution for every particle. This straightforwardly utilize data gotten by sensors when particles evolve. This is likewise an expansion of past approach [33], which does not have the capacity of incorporating the odometry data. Particularly, few circumstances where bad laser feature to localization accessible, this perform superior to past one. The proposal distribution calculation is done correspondingly FastSLAM-2 introduced by Montemerlo [34]. As opposed by FastSLAM-2, do not depend upon landmarks that are predefined also use crude laser range information to obtain exact grid map. Moralez Men'endez [35] gave technique for all the reliable estimate state condition to dynamic framework with precise sensors accessible. Upside of approach is more. Right off the bat, the algorithm will draw the particles with most compelling way. Furthermore, profoundly precise proposal distributions enables for effective sample size with vigorous pointer for choosing resampling. It decreases danger with particle depletion.

High Level Design of Delibot with SLAM Implementation

In product development, High-level plan (HLD) assumes an essential part. It gives whole framework's overview, components that are real prerequisite for the framework will be recognized, and so on. It gives review about design of the project, architecture, system environment and securities/ policies.

II. DESIGN CONSIDERATIONS

Clean partition of the modules is an essential outline thought as it permits extensibility of a module without modifications to alternate modules. Each of the modules should open interfaces to be utilized by the dependent module. Some of assumptions and constraints during implementation of project are as follows: Robot should be intelligent enough to choose best path with minimal distance with training; Wide range of understanding of communication among components to be considered; Detection of object fails if the object is below kinect camera; For better obstacles detection, there should be less background clutter and light condition.

III. DEVELOPMENT METHODS

There are a few software development methods accessible like water-fall model, iterative approach, spiral model, agile methodology and so forth. The project is developed utilizing agile development method. Every day stand-up meetings were directed to keep updates about the advance being made. This specific development enables changes to be fused at any phase of project development, this likewise helps in quick and adaptable response to the progressions [56]. Figure demonstrates the agile development methodology.

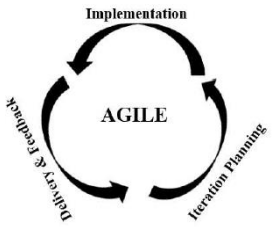

Figure: Agile Development Methodology

Architectural Strategies

For the most part large systems are disintegrated into various sub-systems where each of it performs some related tasks. Architecture design manages identifying the different sub-systems and designing complete system for building up communication

between sub frameworks and controlling them, the output of this procedure is software architecture description.

Programming Language

1. The software of this robot is for the most part implemented utilizing the Python programming language and software frameworks, as Robot Operating System (ROS), Open-CV, etc. ROS is compatible with Python

2. Python is utilized for the designing of the robot and robot's UI. Programming Vision Sensors are done using Python.

3. Python is used to make the robot interactive. Autonomous navigation of the robot is done using ROS and Python.

4. Python is an interpretive language, its prime focus is ease of use.

5. Free libraries to implement basic functionalities are there.

6. Python allows C/C++ code binding, where heavy code part is implemented for avoiding the loss in performance.

Future Plans

Upgrade of the present hardware setup, because of the tight FoV of the Kinect which may cause unreliable navigation. Likewise, framework can be utilized to perform SaR missions, the robot ought to have the capacity to explore and navigate towards victims, and eventually interact with them. Use second Kinect sensor to accomplish a more extensive FoV. This change does not suggest extraordinary expenses and should yield a more secure navigation. Later on, likewise plan to exploit different abilities of the Kinect sensor, for example, handling sound data from its microphone array for people detection. Enhance navigation stack and do movement of TurtleBot faster in react with more flexible

Error Detection and Recovery

Automated discovery of error and recovery from it expands the reliability of the application [35]. Since this is as yet an exploration project and not commercial one which should be deployed, it can't identify errors and recover. However, this can be incorporated into the when it turns into fully fledged utilized by different frameworks for object detection human computer interactions and so on.

Data Storage Management

Data is the fundamental vital piece of this project which is utilized to train the new models for obstacles detection. Data is captured utilizing Kinect camera which would catch almost seventeen frames for every second on average.

Communication Mechanism

Communication between different modules is completed by passing the output of one module to the following module, which utilizes it as input, it processes that information and returns some outcome, result of this module is given to the following and goes on, this can be checked by reviewing intermediate results.

IV. SYSTEM ARCHITECTURE OF SLAM

Control system of the delibot has layered structure and was acknowledged with a couple of computations units. One of the requirements for low level control system is operations in real-time. Hence, time sensitive algorithms are run in microcontroller Kinetis. The control system high level part doesn't have to satisfy hard real time requirements so non realtime interfaces were utilized like an Ethernet or a USB. The fig. 4.2 shows the architecture of control system. The higher layer of control framework was realized in the ROS Indigo system. This environment for development gives mechanisms to nodes of software synchronization and communication. In addition, ROS permits to simple run control algorithms on few computers. Along these lines, some portion of control framework was keep running on Intel platform. Components are realized as nodes which communicate with every others by mechanism called topics. The control framework comprises of following components:

1. Keyboard teleoperation
2. Velocities multiplexer
3. Velocities filter
4. Localization filter
5. User application with rviz
6. Navigation
7. Kinect Openni
8. Laser scan from Kinect
9. Hardware driver

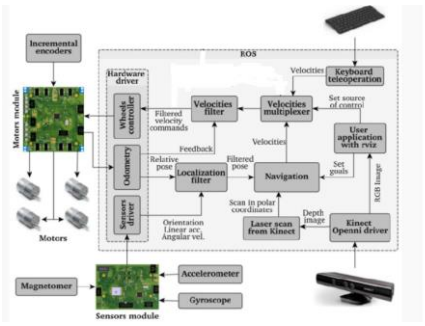

Figure: The Architecture Of High-Level Control System Of Mobile Robot

Keyboard Teleoperation

The component permits to use specific keyboard keys to control the platform. It is realized in Python script where function of keys are determined.

Velocities Multiplexer

The velocities multiplexer component permits to switch between commands sources of velocities for instance manual control or autonomous navigation.

Velocities Filter

The component filters-velocities send to the low level controller. It permits to bound accelerations and velocities of platform of mobile.

Localization Filter

To enhance odometry localization result, it utilized inertial estimation unit (IMU). The information combination based on Extended Kalman Filter (EKF).

User Application With Rviz

The product software permits to control the platform by user. It depends on ROS tool for visualization — rviz. It shows robot map, picture from Kinect RGB camera and gives elements to set autonomous navigation objectives.

Navigation

Reconfigured standard ROS bundle for tasks of navigation.

Kinect Openni

The OpenNI driver for Kinect. It permits to get from Kinect sensor information like RGB picture or depth image. The information are distributes to specific topics.

Laser Scan from Kinect

The software changes Kinect sensor' depth image to 2D laserscan message (sensor_msgs/LaserScan). Also it expel ground from depth image and compensates tilt angle of sensor. The component was distributed as a major aspect of a depth_nav_tools bundle on ROS page.

Hardware Driver

The committed driver for hardware components of robot. It gets information from sensors module, motors controller which publishes them to ROS topics.

V. DATA FLOW DIAGRAMS

Data Flow Diagram (DFD) gives visual portrayal of the flow of data inside a system. It generally goes about as a fundamental step for making an overview of system without diving into details in depth. It gives flow of data for any system or process. The components utilized for DFD are data store, process, and data flow.

DFD Level 0 of SLAM

Single process gives brief depiction of the system of all entities through DFD Level 0. It is the top level of DFD. The complete Data flow diagram level0 of the application is portrayed in Figure which shows the sensors and odometry input modules are given to SLAM . Once these inputs are provided, SLAM creates the map and estimates pose of robot.

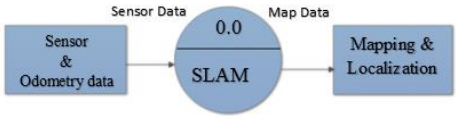

Figure: Level 0 Data Flow Diagram of SLAM

DFD Level 1 of SLAM

DFD Level 1 demonstrates how the whole framework separated into sub systems where every unit manages data-flow out and in via external entity. Level-1 DFD for the proposed framework is given in Figure .

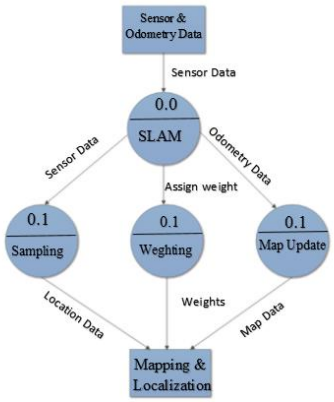

Figure: Level 1 DFD of SLAM

In figure the next level modules are explored. These are a set of independent sub processes with respect to SLAM. The sub processes performs task such as Sampling, Weighting and Map Update for creating the map and estimate the pose of robot.

DFD Level 2 of SLAM

DFD level 2 of SLAM as shown in Fig. is detailed level-diagram further dividing processes of DFD - level 1. The sensors and odometry input modules are given to SLAM. Then sub-processes performs task such as Particle Generation, Data Association and re-sampling which performs sampling, weighting and updation of map. Once these steps are done, SLAM creates the map and estimates pose of robot.

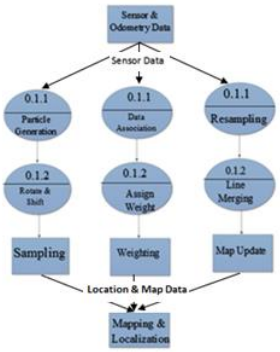

Figure : Level 2 Data Flow Diagram of SLAM

Implementation of Delibot using SLAM

Implementation is an essential stride of SDLC after detailed design of project, amid which programmer codes the application utilizing an appropriate programming language, platform and so on .Before beginning the implementation a few vital choices must be taken with respect to the choice of programming language, platform and so on., a few factors, for example real environment in which this application works, security concerns, expected speed and so on will have its impact.

Programming Language Selection

Programming language made use is discussed below along with reason behind its use.

Python

1. The software of this robot is implemented the Python programming language and ROS is compatible with Python

2. Python is used for designing of robot's UI. Programming Vision Sensors are programmed using Python.

3. Autonomous navigation of robot is done utilizing ROS and Python. Python is utilized to make the robot interactive.

4. Python is used because of its convenience and easy to use. There are number of free libraries to execute essential functionalities.

5. Python decreases time in programming, for example, casting and defining variable types. Python permits bindings with C/C++ code and reduce performance loss.

Platform Selection

Intel launched a minicomputer - Intel NUC. It has a low form factor, is lightweight, and has a decent computing processor with Intel Celeron, Core i3, or Core i5. It supports up to 16 GB of RAM and has incorporated Wi-Fi/Bluetooth. We are picking Intel NUC on the grounds that of its performance, lightweight and ultra small form factor. We are not going for a mainstream board, for example, Raspberry Pi (http://www.raspberrypi.org/) or BeagleBone in light of the fact that we require high processing power for this situation, which can't be given by these. The NUC we are utilizing is Intel DN2820FYKH. Here are the determinations of this PC:

• Intel Celeron Dual Core processor with 2.39 GHz

• 4 GB RAM

• Intel integrated graphics

• 500 GB hard plate

• 12 V supply

• Headphone/ microphone jack

Code Conventions

It is standard practice to embrace some coding standard before the real implementation begins. Since, it will have a great effect later amid integration, testing and deployment stage. One such case for Coding convention is GNU standard. These convention additionally increment the readability of the code. At long last, it helps in understanding the code in the event that it is handover to some other developer.

Naming Conventions

The name utilized for functions ought to be agreeing the functionality they are doing and the module which it part is of. Name of the variable ought to be as per the sort of the data it holds. The constants are in capital letters. Additionally, it is basic to give the relevant names to files. Name of record file to uncover the reason for writing that code. Names have an imperative part inside ROS. Each parameter, node, topic, and service has distinctive name. Architecture takes into account decoupled operation which permits substantial, difficult frameworks for build. ROS underpins command line renaming name, implies compiled software reconfigured in runtime for working at alternate topology. This suggests a comparable center point can be run various conditions, conveying refinement messages to confine subjects.

File Organization

The filesystem was ROS association structure of machine called package have ROS runtime software as node, libraries, third-party software, even datasets. With objective of giving efficient easy functionalities for reusing for a wide range of project. Alongside OO programming, enables package for modules build, work together for achieving end-state. Package regularly take after typical structure which normally have accompanying components: message types, service types, package manifests, executable scripts, headers, runtime processes, and build file. Package gives metadata about a package, for example, the

name, creator, description, dependencies, version, and license information. It have type of messages, that characterize information for messages structure inside ROS. Likewise filesystem have repositories that share a typical control framework. Repositories also packages do ROS as modular framework.

Comments

Comments are added to make the code more readable by including additional data. It is constantly better to include comments as and when it is coded instead of postponing it for future. Points of interest of remarks are as underneath: Used to legitimize decisions; • To portray limitations any ; To determine about the required improvement

Integration of Components

ROS is configured for running on machine by using setup step accessible as. Fig. represents the general ROS integration architecture of our framework. The essential framework operation steps is represented underneath:

1. Issue motor cmd: utilizing console key, front, left/right move is done. The motion_controler node will publish wheel speeds to /motion.

2. Motion activation: on movement cmd, robot motors will be actuated. rosserial_python configure, publish /odometry.

3. Odometry: perusing encoder of wheel utilizing drive model, robot 2D posture estimate was gotten. tf_frames gets odomtry information, publishing frame estimated_pose. Node is created in light of the methods, for publishing frame for/tf.

4. LRF scan: as robot translates with displace determined according to linearUpdate, angularUpdate gmapping parameters individually, another laser publish /scanby gmapping.

5. SLAM: gmapping acquires odometry /tf, and most recent laser uses to correct pose with update mapping. Correct posture publish for correct_pose.

6. Visualization: rviz arrange for map, corrected, estimated pose and most recent laser.

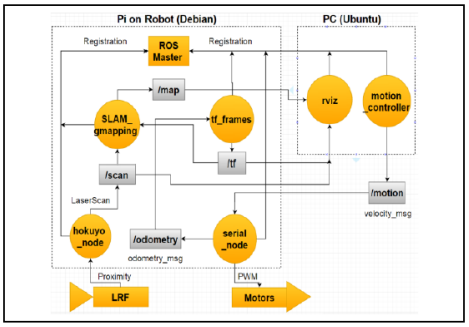

Figure : Integration of SLAM and ROS

Openni Stack

Driver packages of vision sensor, convert over raw RGBD into depth with point cloud using openi_cam and openi_launch. openi_cam publish camera_info information for vision and depth sensor. Depth image have pixel of RGB picture as in Fig .

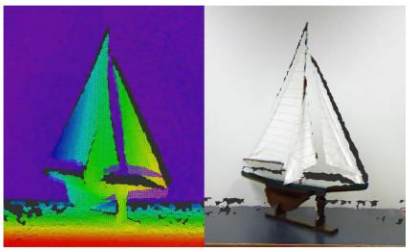

Figure : Left picture caught by vision depth sensor, that demonstrates pixels of most max range set apart in purple, min with red. Other picture shows depth

Commands for download, run openi_cam, openi_launch, found at Fig 6.3. Data publish can be seen utilizing Rviz as in Fig .

```
# Install openni_camera, openni_launch, and openni_tracker
$ sudo apt-get install ros-<ros_distro>-openni-camera ros-<ros_distro>-openni-launch ros-
<ros_distro>-openni-tracker

# Build openni stack
$ cd ~/catkin_ws
$ catkin_make

# Run openni_launch
$ roslaunch openni_launch openni.launch camera:=kinect depth_registration:=true

# Run openni_tracker (in a separate terminal window)
$ rosrun openni_tracker openni_tracker

# View feed from Microsoft Kinect (in a separate terminal window)
$ rosrun rviz rviz
```

Figure : Command used for download, run openni packages to vision sensor

Navigation Stack

The gmapping package inside the navigation stack is in charge of giving SLAM laser. openi_cam gives point-cloud information of vision sensor, depthimg_to_laser is used for gmapping necessities to laser information. depthimg_to_laser changes over a point-cloud information to depth, arranged by sensor_msg/LasrScan.msg as shown at Fig.

```
# Install depthimage_to_laserscan package
$ sudo apt-get install ros-<ros_distro>-depthimage-to-laserscan

# Build depthimage_to_laserscan package
$ cd ~/catkin_ws
$ catkin_make

# Run openni_launch
$ roslaunch openni_launch openni.launch camera:=kinect depth_registration:=true

# Run depthimage_to_laserscan (in a separate terminal window)
$ rosrun depthimage_to_laserscan depthimage_to_laserscan image:=/kinect/depth/image_raw

# View feed from Microsoft Kinect (in a separate terminal window)
$ rosrun rviz rviz
```

Figure : depthimage_to_laserscan usage

The gmapping utilizes a Rao-Blackwellized. For lessen of regular particle depletion related issue gmapping utilizes resampling [61]. gmapping utilized 2-D grid technique for map build. GMapping likewise uses scan matching, contrasting current laser checks with past laser scans keeping in mind the end goal to lessen odometry err. slam_gmaping find obstacle, with low probability remove ocupancy grid. The

commands used keeping in mind the end goal of using gmapping appeared at Fig .

```
# Install gmapping package
$ sudo apt-get install ros-<ros_distro>-gmapping ros-<ros_distro>-slam-gmapping

# Build gmapping package
$ cd ~/catkin_ws
$ catkin_make

# Run slam_gmapping node
$ roslaunch gmapping slam_gmapping scan:=scan odom_frame:=odom
```

Figure : Command usage of gmapping

Autonomous Navigation

mov_base bundle is navigation stack part. Use global also local costmap via costmap_2d to accomplish global navigational task. A costmap is a kind of occupancy grid, where every costmap cell exclusively set apart free, unknown, occupied additionally has 0 and 254 value of cost. The depiction of costmaps, is in Figure cell set apart as red obstacles, blue is obstacle of robot orientation, grey thought is space. Keep away from obstacles, robot not to cross blue cell. The global_planner package can use a few diverse path plan, for example, Dijkstra, or A* algorithm.

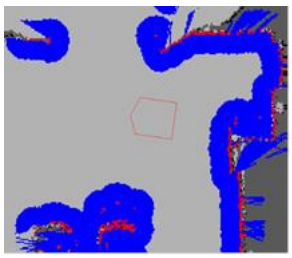

Figure : Portrayal of a costmap,

Robot is wirelessly driven all through environment with openi_cam with RGBD pictures of vision sensor. openi_cam publish camera data through /kinect/depth/img_raw, /depthimg_to_laser subscribe for point cloud information sensr_msg/LasrScn.msg publish by means of /scan shown in Fig . slam_gmapping node use scan matching and particle filtering so as to direct SLAM.

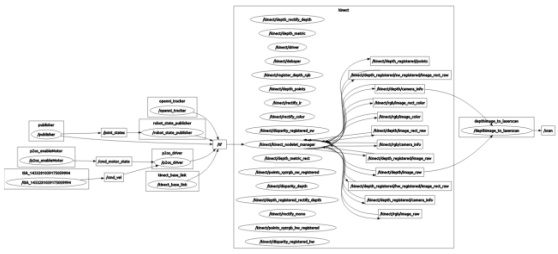

Figure : Communication amongst node with topic using rqt_graphs

As mov_base won't give localization, AMCL (Adaptive Monte Carlo Localization) is used. It utilizes Monte Carlo localization that utilizes particle filter for tracking robot pose with existing map. With a specific end goal for autonomous navigation and SLAM with environment, mov_base also slam_gmaping run all while. In spite of the issues with odometry mistakes, the robot could effectively autonomously navigate to goal in dynamic and unknown environment while leading SLAM. The commands for achieving program cycle as fig step is finished with similar request at terminal window.

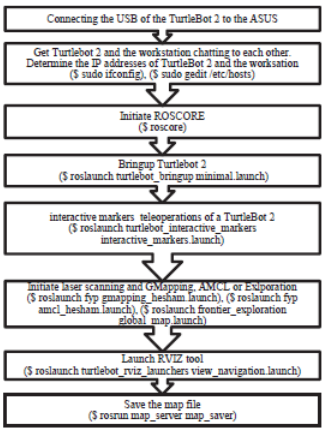

FIG. 4 ALL ROS COMMANDS LISTED IN ORDER TO RUN AS DESCRIBED IN PROGRAM LIFECYCLE

Figure : ROS commands for program cycle

Experimental Analysis and Results of Delibot with SLAM

With a specific end goal to illustrate impact of systematic strategy done using controlled conditions named experiment. Analysis find effect of input upon output, additionally find objective input for required output.

Evaluation Metric

These are for the most part used to evaluate development of software, metrics are utilized for finding product quality. Numerous metrics are there for different process. Inquires conveyed for discovering relationship between metrics and issues showing product quality. Following are the metrics considered to evaluate the SLAM of delibot,

1. Estimation of the pose of the robot with the information measured by sensors on the robot.

2. Estimation of landmark positions.

3. Correction of the position of the robot amid each time step using landmark data acquired by sensors

4. Map of environment.

Experimental Dataset

The data set used are the odometry data of the robot, ultrasonic sensors data and the Vision camera data which are fed dynamically when the robot is first trained. All these data are used by the robot to estimate the position of it and the mapping of environment. And 5 particles was used initially which gave resampling limit of about 0.5 with step among laser checks processed. In any case, these

values may somewhat differ as environment length to build map.

Performance Analysis of SLAM

The FastSLAM framework is utilized to reduce computational complexity of the SLAM issue. Updating posterior take O(MlogK) time, particle no. is M also landmarks number is K. Here analysis of framework performance got from tune parameters of gmapping is presented. The analysis concentrated on tuning resampling threshold, particle number with step measure among progressive laser scan. Moreover, the created maps for each situation are likewise assessed visually for map exactness.

1. Number of Particles

Despite the fact that realized increasing particles number adds for improving SLAM precision, presents high CPU load also memory utilization as in Table .

TABLE . Impact as Increase RBPF Particles Number

Particles	CPU Load	Memory (MB)
5	5%	5
15	17%	54
30	17%	54
50	22%	56

It demonstrates gmapping system adequately decreases particles number keeping map accurate and resampling threshold legitimately. Obviously, should remember particles required for deliver of good outcomes depends upon environment mapped length.

LinearUpdate Step

Gmapping permits modifying processing observations using parameters of angular and linear update. Making parameters for large value diminishes CPU caused while preparing process of new scanned information. Then again, such setting may bring about poor developed maps because algorithm miss vital environment feature. Algorithm turns out to be highly uncertain about right position of robot, when it's static. Result is recorded in Table underneath.

TABLE Effects on Increasing Laser Process Step

Step (m)	CPU Load	Memory (MB)
0.2	20%	39
0.4	17%	31
0.7	16%	28

Resampling Threshold

Predetermined limit of Neff measurement is used for resampling. Solid resampling impact "particles depletion" [62], which is chosen for tuning resampling threshold. This have the capacity for catching impact of particles depletion, furthermore assess impact on algorithm performance.

TABLE Impacts of Tuning The Resampling Threshold

Resample Threshold	Particles	CPU Load	Memory (MB)
0.1	30	15%	46
0.1	5	12%	42
1	30	23%	58
1	5	22%	54

The outcomes acquired in Table , demonstrate assumption with respect to the impact of high resample edge. As gets high, CPU also memory consumption decrease with increase generally. Outcomes are portrayed in Fig which represents that an increase in number of landmarks gently decreases error in map and the robot posture. Outlines of 8.1, the bars relate to 95% certainty intervals. Substantial landmark decreases localize error. Red dotted line is the landmark position error and blue solid line is the robot position error.

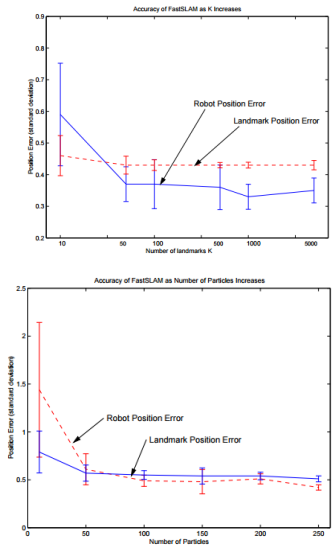

Figure : Accuracy of the FastSLAM calculation as an element of (a) landmark no. (b) particles no.

Inference from the Result

For performance regarding mapping and computation, 5 particles gave resampling limit of about 0.5 with step among laser checks processed adequately little. In any case, these values may somewhat differ as environment length to build map. Speed of robot influences map accuracy. While the robot should be operated adequately slowly to effectively distinguish corner, quicker when goes in one direction. In view of that, speed strategy created for outcomes change speed to diminish the general mapping time. FastSLAM landmarks estimate need O(MlogK), M particles also K landmarks. This is rather than the O(K2) Kalman- filter way of dealing. Trial show FastSLAM build map of greater number of landmarks than past strategies. They additionally exhibit that under specific conditions, few particles do irrespective of landmarks.

VI. CONCLUSIONS

The SLAM is concerned about buildingan intelligent robot that identify its position in an unknown environment, while incrementally build the map of environment it is. The processof integration of ROS with platform of mobile robot along with vision-depth sensors leading SLAM and autonomous navigation within dynamic and unknown environment is researched. A ROS fundamental with hardware capacities, to be specific mobile robot along with RGBD sensor was investigated. Download, configure, testof mobile robot, SLAM, vision sensor and autonomous navigation was done using ROS package. SLAM part was tested when wirelessly tele-operated the mobile robot via keyboard, making environment map. Then, component of the autonomous navigation was explored utilizing already existing map, robot avoiding static and dynamic obstructions to accomplish target. Lastly, robot accomplished targetas itidentify and map obstructions and package parameters was changed to enhance framework. SLAM and autonomous navigation utilizing ROS with moderate depth sensor, for example, the Microsoft Kinect was accomplished. ROS, with its object-oriented programming and modular architecture, gave an easy and robust system of robot platform, map construction packages, vision sensors, and navigation controllers autonomously to work for achieving target. In real time maps were built using gmapping node via autonomous navigation along with teleoperation. And move_base package accepted data by map. Gmapping with scanned information, created costmap, found global and local paths, with controlling robot via dynamic location to achieve target. In ROS, messages publish for topics of right format, the outcomes acquired can be recreated using a wide range of mobile robot platforms, and different types of depth sensors.

VII. ACKNOWLEDGEMENTS

The author thanks Dr.S.Sridhar, Professor and Director, Cognitive & Central Computing, R.V.College of Engineering, Bangalore, India for communicating this article to this Journal for publication after modifications. The author also thanks Dr.G.Shobha, Professor and Head, Department of CSE, R.V.College of Engineering, Bangalore, India for the support given.

VIII. REFERENCES

[1] J. Neira and J.D. Tard´os. Data association in stochastic mapping using the joint compatibility test. IEEE Transactions on Robotics and Automation, pp.890–897, USA 2001.

[2] B. Borovac, "Humanoid Robotics: What We Really Need - Just Machines, or Something More" in Intelligent Systems and Informatics (SISY), 2012 IEEE 10th Jubilee International Symposium on, Subotica,pp. 190-200, Serbia, 2012.

[3] H. Durrant-Whyte, T. Bailey, "Simultaneous localization and mapping (SLAM): Part II", IEEE Robotics & Automation Magazine, vol. 13, pp. 108-117, Malaysia 2006.

[4] J. Li, L. Cheng, H. Wu, L. Xiong, D. Wang, "An overview of the simultaneous localization and mapping on mobile robot", Modelling Identification & Control (ICMIC) 2012 Proceedings of International Conference on, pp. 358-364, Italy 2012.

[5] H. Durrant-Whyte, T. Bailey, "Simultaneous localization and mapping (SLAM): Part I", IEEE Robotics & Automation Magazine, vol. 13, pp. 99-110, USA 2006.

[6] A. Pascal, J. Kuhn, "Simultaneous localization and mapping (SLAM) using the extended kalman filter", In Proc. of the American Control Conference, pp. 1628–1632, Seattle, WA, USA, 2013.

[7] F. Pirahansiah, S. N. H. Sheikh Abdullah, S. Sahran, "Simultaneous Localization And Mapping Trends And Humanoid Robot Linkages", Asia-Pacific Journal of Information Technology and Multimedia, vol. 2, pp. 1466-1480, Asia 2013.

[8] G. Dissanayake, S. Huang, Z. Wang, R. Ranasinghe, "A review of recent developments in Simultaneous Localization and Mapping", 6th International Conference on Industrial and Information Systems (ICIIS), pp. 477-482, USA 2011.

[9] H.P. Moravec. Sensor fusion in certainty grids for mobile robots. AI Magazine, pp. 61–74, USA 1988.

[10] J.J. Leonard and H.F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. IEEE Transactions on Robotics and Automation, pp.376–382, 1991,Siena, Italy.

[11] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial realtionships in rob11otics. In I. Cox and G. Wilfong, editors, Autonomous Robot Vehicles, pages 167–193. Springer Verlag, 1990.

[12] B. T. Y. Loh, Improved Map Generation By Addition of Gaussian Noise For Indoor Mobile Robot SLAM Using ROS, Selangor, Malaysia: Collage of Engineering,

University Tenaga Nasional,pp. 567-589, Malaysia 2013.

[13] U. Frese and G. Hirzinger. Simultaneous localization and mapping - a discussion. In Proc. of the IJCAI Workshop on Reasoning with Uncertainty in Robotics, pp. 17–26, Seattle, WA, USA, 2001.

[14] S. Julier, J. Uhlmann, and H. Durrant-Whyte. A new approach for filtering nonlinear systems. In Proc. of the American Control Conference, pp. 1628–1632, Seattle, WA, USA, 1995.

[15] T. Duckett, S. Marsland, and J. Shapiro. Fast, on-line learning of globally consistent maps. Journal of Autonomous Robots, pp.287 – 300, Italy 2002.

[16] U. Frese, P. Larsson, and T. Duckett. A multilevel relaxation algorithm for simultaneous localisation and mapping. IEEE Transactions on Robotics, pp.1–12, USA 2005.

[17] J.-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In Proc. of the IEEE Int. Symposium on Computational Intelligence in Robotics and Automation (CIRA), pp. 318–325, Monterey, CA, USA, 1999.

[18] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. Journal of Autonomous Robots, pp. 333–349, Spain 1997.

[19] D. H¨ahnel, W. Burgard, B. Wegbreit, and S. Thrun. Towards lazy data association in slam. In Proc. of the Int. Symposium of Robotics Research (ISRR), pp. 421–431, Siena, Italy, 2003.

[20] R. Eustice, H. Singh, and J.J. Leonard. Exactly sparse delayed-state filters. In Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), pp. 2428–2435, Barcelona, Spain, 2005.

[21] M.A. Paskin. Thin junction tree filters for simultaneous localization and mapping. In Proc. of the Int. Conf. on Artificial Intelligence (IJCAI), pp. 1157–1164, Acapulco, Mexico, 2003.

[22] J. Folkesson, P. Jensfelt, and H. Christensen. Vision SLAM in the measurement subspace. In Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), pp. 325–330, USA April 2005.

[23] F. Dellaert. Square Root SAM. In Proc. of Robotics: Science and Systems (RSS), pp. 177–184, Cambridge, MA, USA, 2005.

[24] A. Doucet, J.F.G. de Freitas, K. Murphy, and S. Russel. Rao-Blackwellizedpartcile filtering for dynamic bayesian networks. In Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI), pp. 176–183, Stanford, CA, USA, 2000.

[25] K. Murphy. Bayesian map learning in dynamic environments. In Proc. of the Conf. on Neural Information Processing Systems (NIPS), pp. 1015–1021, Denver, CO, USA, 1999.

[26] M. Montemerlo and S. Thrun. Simultaneous localization and mapping with unknown data association using FastSLAM. In Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), pp. 1985–1991, Taipei, Taiwan, 2003.

[27] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to simultaneous localization and mapping. In Proc. of the National Conference on Artificial Intelligence (AAAI), pp. 593–598, Edmonton, Canada, 2002.

[28] Andrew Howard. Multi-robot simultaneous localization and mapping using particle filters. In Robotics: Science and Systems, pp. 201–208, Cambridge, MA, USA, 2005.

[29] M. Bosse, P.M. Newman, J.J. Leonard, and S. Teller. An ALTAS framework for scalable mapping. In Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), pp. 1899–1906, Taipei, Taiwan, 2003.

[30] J. Modayil, P. Beeson, and B. Kuipers. Using the topological skeleton for scalable global metrical map-building. In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), pp. 1530–1536, Sendai, Japan, 2004.

[31] B. Lisien, D. Silver D. Morales, G. Kantor, I.M. Rekleitis, and H. Choset. Hierarchical simultaneous localization and mapping. In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), pp. 448–453, Las Vegas, NV, USA, 2003.

[32] D. H¨ahnel, W. Burgard, D. Fox, and S. Thrun. An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), pp. 206–211, Las Vegas, NV, USA, 2003.

[33] G. Grisetti, C. Stachniss, and W. Burgard. Improving grid-based slam with Rao-Blackwellized particle filters by adaptive

proposals and selective resampling. In Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), pp. 2443–2448, Barcelona, Spain, 2005.

[34] M. Montemerlo, S. Thrun D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In Proc. of the Int. Conf. on Artificial Intelligence (IJCAI), pp. 1151–1156, Acapulco, Mexico, 2003.

[35] R. Morales-Men´endez, N. de Freitas, and D. Poole. Real-time monitoring of complex industrial processes with particle filters. In Proc. of the Conf. on Neural Information Processing Systems (NIPS), pp. 1433–1440, Vancover, Canada, 2002.

[36] R. C. Smith, P. Cheeseman, "On the representation and estimation of spatial uncertainty", *The international journal of Robotics Research*, vol. 5, pp. 56-68, Italy 1986.

[37] M. R. Naminski, "An Analysis of Simultaneous Localization and Mapping (SLAM) Algorithms", *Mathematics Statistics and Computer Science Honors Projects*, pp. 29Thaiwan2013

[38] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using Kinectstyle depth cameras for dense 3D modeling of indoor environments. International Journalof Robotics Research (IJRR), pp.647–663, USA April 2012.

[39] Felix Endres, Jürgen Hess, Nikolas Engelhard, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. An evaluation of the RGB-D SLAM system. In Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 1691–1696, Japan 2012.

[40] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. Proc. of the IEEE Int. Conf. on Robotics andAutomation (ICRA), pp. 3607–3613, USA May 2011.

[41] HanspeterPfister, MattiasZwicker, Jeroen van Baar, and Markus Gross. Surfels: Surface elements as rendering primitives. ACM Transactions on Graphics (Proc. of SIGGRAPH), Italy 2000.

[42] MWM GaminiDissanayake, Paul Newman, Steve Clark, Hugh F Durrant-Whyte, and Michael Csorba. A solution to the Simultaneous Localization and Mapping (SLAM) problem. Robotics and Automation, IEEE Transactions on, pp.229–241, Taiwan 2001.

[43] J.Machado Santos, D. Portugal, and R.P. Rocha. An evaluation of 2D SLAM techniques available in Robot Operating System (ROS). Safety, Security, and Rescue Robotics(SSRR), 2013 IEEE International Symposium on, pp. 1–6, USA Oct 2013.

[44] Stefan Kohlbrecher, Oskar Von Stryk, Johannes Meyer, and Uwe Klingauf. A flexible and scalable SLAM system with full 3D motion estimation. In Proceeding of Safety,Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on, pp. 155–160, Italy 2011.

[45] Regis Vincent, Benson Limketkai, and Michael Eriksen. Comparison of indoor robot localization techniques in the absence of GPS. SPIE Defense, Security, and Sensing, pp. 76641–76641, Germany 2010.

[46] Giorgio Grisetti, CyrillStachniss, and Wolfram Burgard. Improved techniques for grid mapping with Rao-Blackwellized particle filters. IEEE Transactions on Robotics, pp.23-28, USA 2007.

[47] KouroshKhoshelham and Sander Oude Elberink. Accuracy and resolution of Kinect depth data for indoor mapping applications. Sensors (Basel), pp.1437–1454, Germany 2012.

[48] Michal Tolgyessy and Peter Hubinsky. The Kinect sensor in robotics education. Proceedings of 2nd International Conference on Robotics in Education (RiE 2011), pp.143–146, Taiwan 2011.

[49] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using Kinectstyle depth cameras for dense 3D modeling of indoor environments. International Journalof Robotics Research (IJRR), pp.647–663, USA April 2012.

[50] C.C. Chen L. Xia and J. K. Aggarwal. Human detection using depth information by Kinect. Workshop on Human Activity Understanding from 3D Data in conjunction with CVPR (HAU3D), USA 2011.

[51] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. ROS: an open-source robot operating system. ICRA Workshop on Open Source Software,Italy 2009.

[52] I. Dryanovski, R.G. Valenti, and Jizhong Xiao. Fast visual odometry and mapping from RGB-D data. Pp. 2305–2310, Taiwan May 2013.

[53] Brian Yamauchi. A frontier-based approach for autonomous exploration. Computational Intelligence in Robotics and Automation 1997,IEEE International Symposium on, pp.146–151, USA 1997.

[54] Pressman, Roger (2010). Software Engineering: A Practitioner's Approach. Boston: McGraw Hill

[55] "IEEE Recommended Practice for Software Requirements Specifications", IEEE Std, 1998, Italy pp.1-40

[56] S. Ken, "Scrum development process". in Proceedings of the Workshop on Business Object Design and Implementation at the 10th Annual Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'95), USA 1995

[57] T. Foote, "ROS," Willow Garage, 27 January 2012. [Online]. Available: http://www.ros.org/wiki/pointcloud_to_laserscan. [AccessedFebruary 2017].

[58] C. Stachniss, U. Frese and G. Grisetti, "OpenSLAM," OpenSLAM Team, January 2007. [Online]. Available: http://openslam.org/. [Accessed February 2017].

[59] D. Hahnel, W. Burgard, D. Fox and S. Thrun, "An Efficient FastSLAM Algorithm for Generating Maps of Large-Scale Cyclic Environments from Raw Laser Range Measurements", in Proc. of the 2003 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), pp.206-211, USA 2013.

[60] G. Grisetti, C. Stachniss and W. Burgar, "Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling", in Proc., IEEE International Conference on Robotics and Automation (ICRA), pp.2432-2437, USA 2005,.

[61] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using Kinectstyle depth cameras for dense 3D modeling of indoor environments. International Journalof Robotics Research (IJRR), pp.647–663, Italy April 2012.

[62] G. Grisetti, C. Stachniss, and W. Burgard, "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters," IEEE Trans. Robot., vol. 23, no. 1, pp. 34–46, USA Feb.2007

