iJRASET
International Journal For Research in Applied Science and Engineering Technology



# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

**Volume:** 10 **Issue:** VI **Month of publication:** June 2022

**DOI:** https://doi.org/10.22214/ijraset.2022.44316

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 10 Issue VI June 2022- Available at www.ijraset.com*

# Autonomous Navigation of Indoor Service Robot with Object Recognition using Deep Learning

Abhijith S[1], Dr. Sreeja S[2]
[1]*Student, Department of Electronics & Communication Engineering, College of Engineering Trivandrum*
[2]*Assistant Professor, Department of Electrical Engineering, College of Engineering Trivandrum*

***Abstract:*** ***Indoor Service robots are becoming popular day by day. The task of service robot is to perform some predefined tasks originally done by human in houses, schools, offices etc. This paper proposes an autonomous indoor service robot system integrated with multi-sensors such as Lidar and Kinect for the smooth functioning of indoor service robots. Firstly, a map of the environment is created using Simultaneous Localization and Mapping (SLAM), a commonly used technique for creating map and positioning the robot at the same. In this paper different Lidar-based SLAM are compared and the best SLAM is chosen for creating 3D map for this system. For autonomous navigation, path planning plays a key role for fulfilling the task. A* algorithm is incorporated with this system to plan a global path. But global path planning algorithm fails to complete its task if any new or dynamic objects come to its path of navigation. To overcome such situations, a local path planning algorithm called Dynamic Window Approach (DWA) is added to this system. Another inevitable element in autonomous navigation is localization of robots in the map. Adaptive Monte Carlo Localization (AMCL) is the technique applied in the proposed system to localize the robot for the effective navigation of indoor service robot. To enhance the performance of service robot, YOLOv5 model is integrated with this system for real time object detection. The entire system is simulated in an open-source framework called Robot Operating System (ROS).***

***Keywords: SLAM, AMCL, A* algorithm, DWA, YOLOv5***

## I. INTRODUCTION

Indoor service robot delivers many useful services for organizations and humans in various sectors. With the advances of artificial intelligence technology, indoor service robots have been growing rapidly. Today, in many sectors, service robots replaced human labour. Many studies have been carried out on how to improve the performance of service robots.

A commonly used technology called Simultaneous Localization and Mapping (SLAM) finds application in indoor service robots for constructing the map of an environment and estimating its own position at the same time, Zhang et al. [1] analysed three different 2D Lidar-based SLAM algorithm in indoor rescue environment along with path planning algorithms. At present, very rare systems combined SLAM and path planning algorithm for the autonomous indoor navigation of service robots. The integration of SLAM with localization and path planning is a challenging task in many situations. Zhang et al. [2] proposed an intelligent hotel service robot based on Robot Operating System (ROS) integrated with Gmapping and D* path planning technique. For any autonomous robot navigation, the ability of localization of robot is an inevitable factor. The problem of autonomous navigation and location is solved in [3] by proposing an efficient Adaptive Monte Carlo Localization (AMCL) method to align the map built by a multi-robot system.

Object recognition capability of robots is a major area of research in the field of robotics. Deep learning object detection algorithms such as You Only Look Once (YOLO), Single Shot Detector (SSD), Faster R-CNN etc have been used in many robotic applications. In [4,5,6,7] object detection using pre-trained Convolutional Neural Networks (CNNs) integrated with visual SLAM to enhance the robotic capabilities. In this paper, three commonly used Lidar-based SLAM algorithms are evaluated in simulation environment. The map constructed with SLAM algorithm is tested with A* algorithm for global path planning and Dynamic Window Approach (DWA) technique for local path planning. The effectiveness of combining SLAM technique with AMCL localization, A* path planning, DWA algorithm and YOLOv5 object detection in autonomous navigation of ROS robot is also studied in this paper.

This paper is organized as follows: Section II describes the system structure; Different mapping techniques are discussed in section III; Section IV explains the localization method; Navigation system is presented in section V; path planning algorithms for navigation is briefly given in section VI; section VII discusses the object detection algorithms used in the proposed system; The conclusion is presented in section VIII.

©IJRASET: All Rights are Reserved 

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 10 Issue VI June 2022- Available at www.ijraset.com*

## II. SYSTEM STRUCTURE

The block diagram of the proposed system is shown in Figure 1. At first, the map of the environment is constructed using SLAM algorithm. To create a map of an unknown environment, the robot has to navigate through the given environment once. The Teleoperation package in ROS fulfils the task of driving the robot manually by accepting keyboard inputs from a remote computer. In this project, hector SLAM is implemented to construct the 2D map of the given environment. It takes lidar scan as input to make the map. Once a perfect map is obtained, navigation of robot to a location is possible only if the robot is correctly localised in the map. AMCL algorithm is used for localization in the proposed system. The navigation stack in ROS which enables autonomous navigation of robot makes use of odometry data and map for the navigation of robot. Also the input location given by user is taken from RVIZ and sends to Navigation stack with the help of goal controller. Then the global path planner in the Navigation stack finds a path from the current location to the goal location. The proposed system uses A* algorithm to obtain an optimal path. In order to handle dynamic or new objects in the path of navigation, a local path planning algorithm called DWA is also incorporated with the system. The Base controller package accepts velocity commands from the Navigation stack and provides appropriate signals to drive the robot to accomplish the task of autonomous navigation of robots in indoor environment. An object detection module is also integrated with the system to identify the objects in the path of navigation of the robot. A model that combines the results of different YOLOv5 models is used in the proposed system for object recognition.

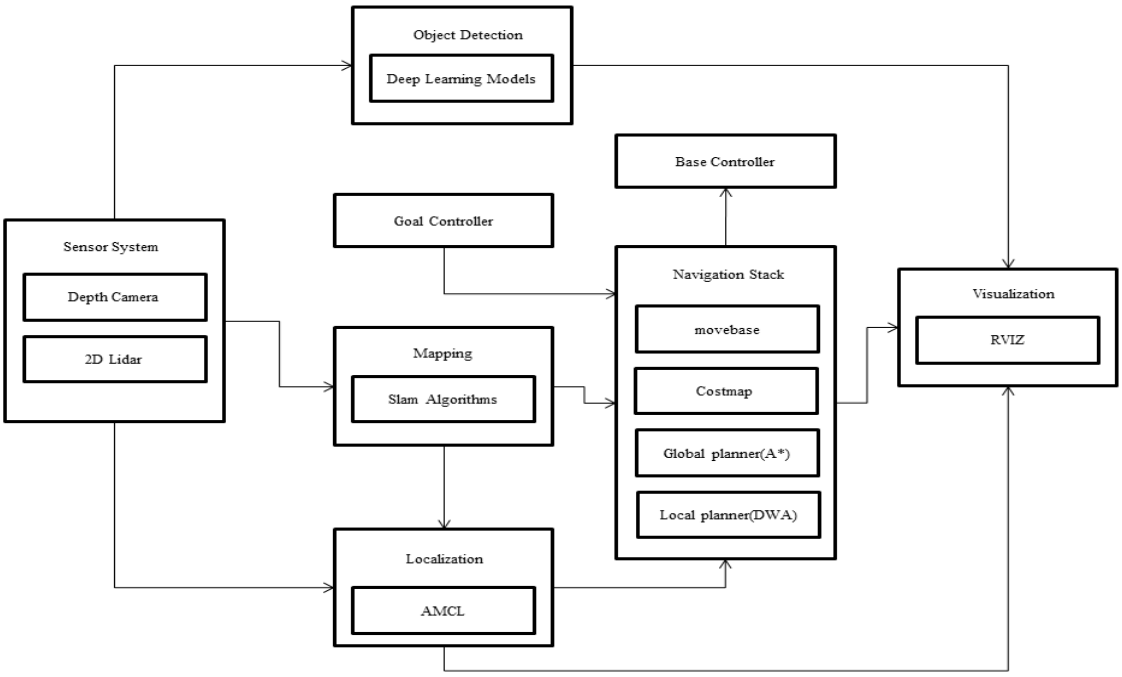



Figure 1: Block diagram of the proposed system

## III. MAPPING

Mapping is a significant step for autonomous navigation robots to do the tasks of indoor service robots. Simultaneous Localization and Mapping (SLAM) is a widely used technology that builds the map of the environment and locates the robot in that map simultaneously.

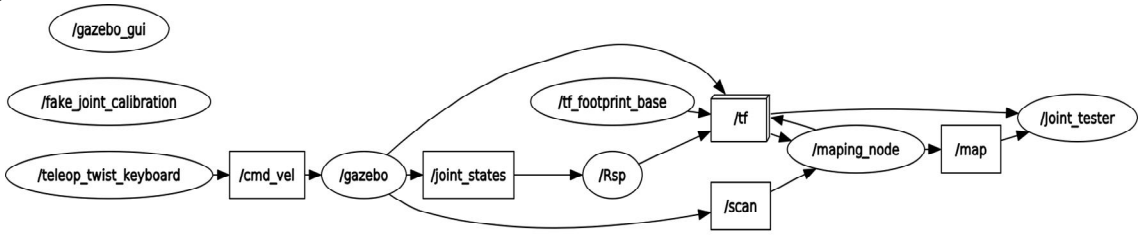



Figure 2: rqt graph for mapping an environment in ROS

©IJRASET: All Rights are Reserved 

*A. SLAM Algorithms*

The most common types of SLAM algorithms are Visual SLAM and Lidar SLAM. The visual SLAM uses the visual input from a camera to construct the map of an unknown environment, whereas Lidar SLAM makes use of laser sensor and odometry to create the map of the surroundings. In this project, lidar based SLAM algorithm is used for mapping. Different types of lidar SLAM algorithms are available. This paper discusses and compares four commonly used lidar SLAM algorithms namely, Gmapping, Hector SLAM, Karto SLAM and Cartographer. The rqt graph for mapping an environment in ROS is given in Figure 2. The /maping_node can be any one of the above mentioned SLAM.

1) *Gmapping*: Gmapping is widely used laser-based SLAM algorithm for creating grid maps. It uses Rao-Blackwellized Particle filter method for constructing the map of a given environment. Sensor data from lidar and odometry from IMU are the inputs used for Gmaping. In this method, a large number of particles are involved in creating a map. Each particle carries an individual map of the environment based on the input data. The four importance steps in Gmapping are sampling, weights calculation, resampling and map updating.

- Sampling: Given a proposal distribution, the particles from previous generation are first sampled. These particles are then improved using the recent observations. Finally, new particles and proposal distributions are generated.
- Weight calculation: As there are many particles, it is important to calculate the weight of each particle to determine the importance of that particle. The weight, $w_t^{(i)}$ is calculated using the equation 1.

$$w_t^{(i)} = \frac{p(x_{1:t}^{(i)}|z_{1:t}, u_{1:t})}{\pi(x_{1:t}^{(i)}|z_{1:t}, u_{1:t-1})} \quad (1)$$

where $z_{1:t}$ is the sensor data, $u_{1:t}$ is the odometry data, $x_{1:t}$ is the robot pose, $p(x_{1:t}^{(i)}|z_{1:t}, u_{1:t})$ denotes the positioning problem and $\pi(x_{1:t}^{(i)}|z_{1:t}, u_{1:t-1})$ is the proposal distribution.

- Resampling: In this step, particles with lower weights and discarded. Also some resampled particles are added keeping the same total number particles.
- Map updating: The map is updated using the pose represented by each particle and the current observation. The update equation is given by equation 2.

$$w_t^{(i)} = w_{t-1}^{(i)} \eta \frac{p(z_t|x_{1:t}^{(i)}, z_{1:t-1}) p(x_t^{(i)}|x_{1:t-1}^{(i)}, u_{1:t-1})}{\pi(x_{1:t}^{(i)}|z_{1:t}, u_{1:t-1})} \quad (2)$$

where η is the normalization factor.

The gmapping package in ROS provides a node called slam_gmapping, which builds a map using lidar scan and odometry. Figure 3 shows the environment created using Gazebo. Figure 4 shows the mapping of environment using Gmapping. The final map obtained using gmap is shown in Figure 5.

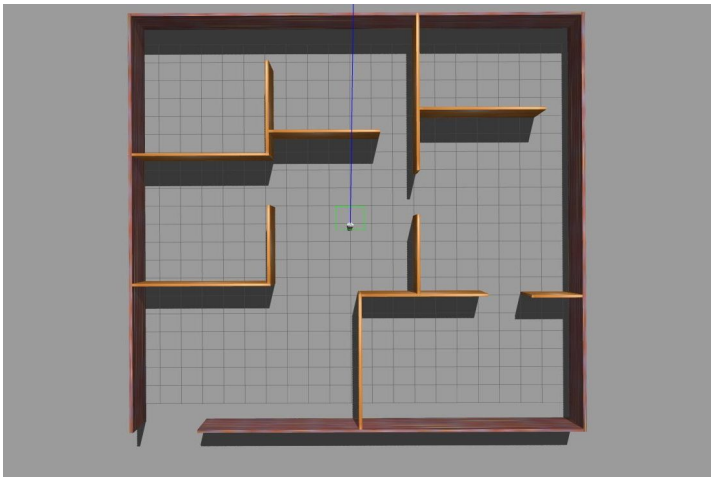


Figure 3: Environment created in Gazebo

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 10 Issue VI June 2022- Available at www.ijraset.com*

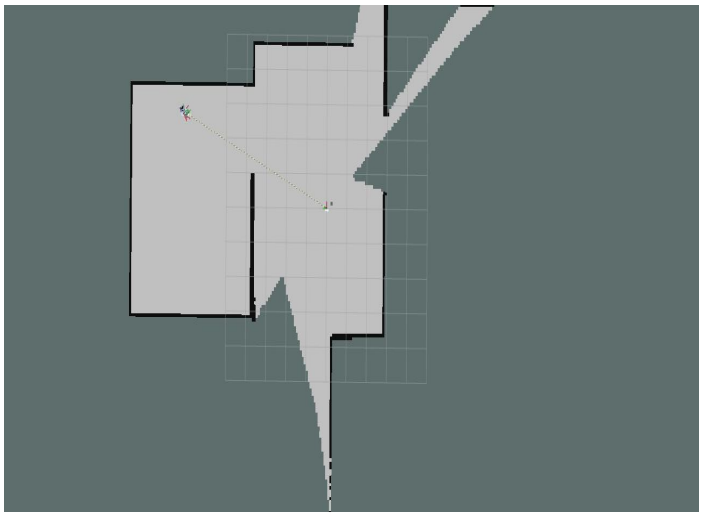


Figure 4: Mapping the environment using Gmapping

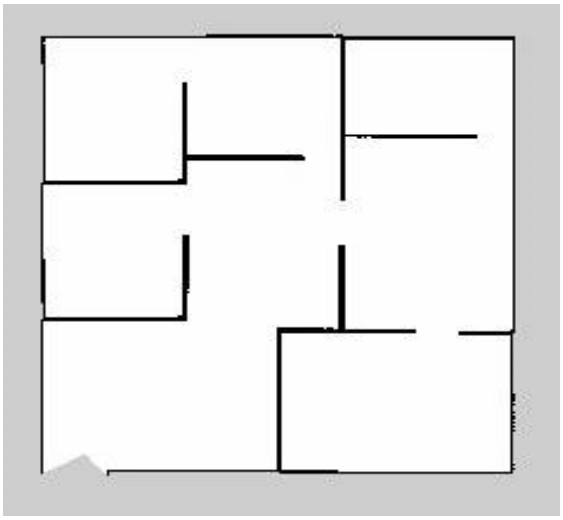


Figure 5: Final map obtained using Gmapping

2) *Hector SLAM*: Hector SLAM is also a lidar-based SLAM. This algorithm uses lidar scan and priori map as input to build the map of an unknown environment. Hector SLAM does not require any odometry data mapping. Hector SLAM makes use of scan matching algorithm to determine the distance translation and rotation robot between two scans. The Gaussian-Newton iteration formula is also used to obtain the optimal pose of the robot in the map. In ROS, hector_SLAM package is available to implement hector SLAM. The Figure 6 shows the mapping of the given unknown environment using hector_SLAM package in ROS. The final map constructed using hector mapping is shown in Figure 7.

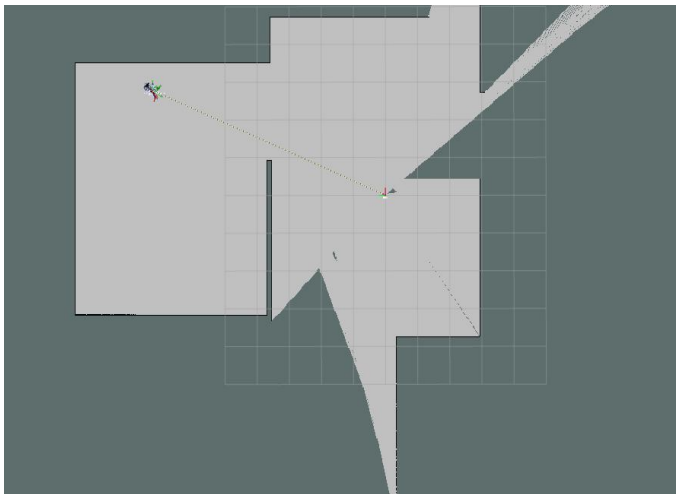


Figure 6: Mapping the environment using Hector SLAM

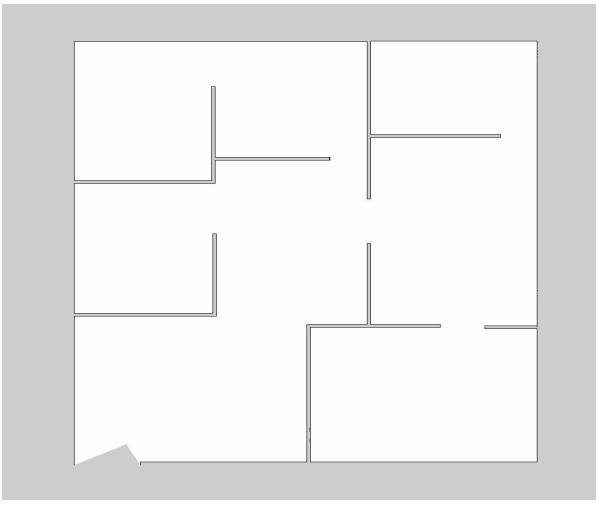


Figure 7: Final map obtained using Hector SLAM

3) *Karto SLAM*: Karto SLAM is a graph optimization-based SLAM algorithm. This algorithm is very useful in cases where the environment to be mapped is very large. As it is based on graph optimization, a diagram with nodes is created for the environment. Each node in the diagram represents the pose of the robot base on its trajectory and sensor data. Edges connecting the nodes represent the movement between positions immediately following each other. When a new node is added, the map is updated by estimating a spatial optimal node configuration. The package used for implementing karto SLAM in ROS is slam_karto. Figure 8 shows the mapping of the environment using karo SLAM. The final map is shown in Figure 9.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

4) *Cartographer SLAM*: Cartographer SLAM is also a graph optimization algorithm. It is a real-time indoor SLAM developed by Google. Cartographer SLAM has the advantage of avoiding the interference of dynamic objects in the environment during mapping. The Google open source SLAM tool is composed of two parts: Cartographer and Cartogra-pher_ROS. The task of Cartographer is to process the data from Lidar, IMU, and odometers to build the map of the given environment.

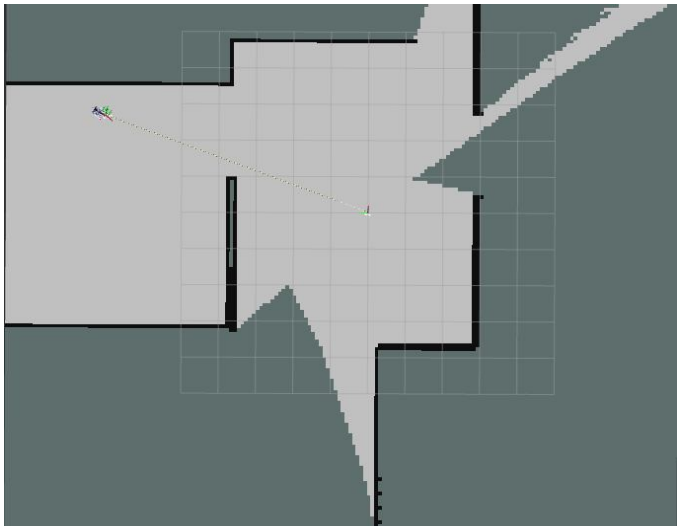


Figure 8: Mapping the environment using Karto SLAM

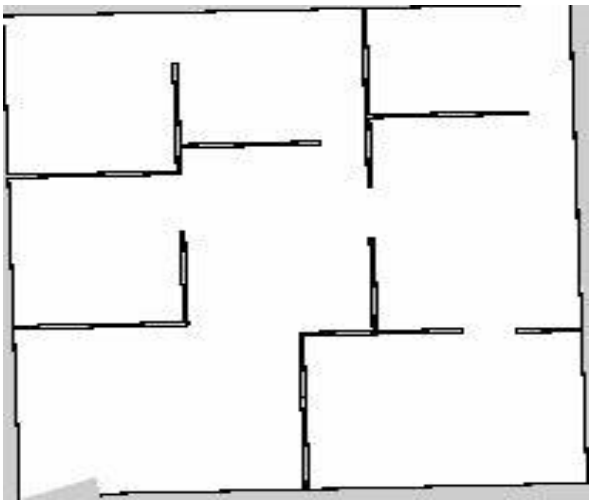


Figure 9: Final map obtained using Karto SLAM

The Cartographer_ROS then converts the acquired sensor data through ROS communication mechanism into the Cartographer format for processing by Cartographer. The Cartographer processing result is released for display or storage.
The package used for cartographer SLAM is cartographer. The mapping using cartographer SLAM is shown in Figure 10. The Figure 11 shows the final map created using cartographer SLAM.

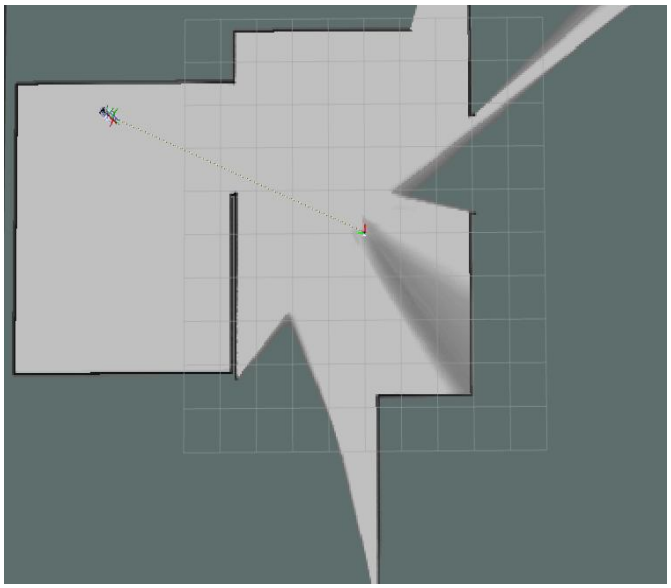


Figure 10: Mapping the environment using Cartographer SLAM

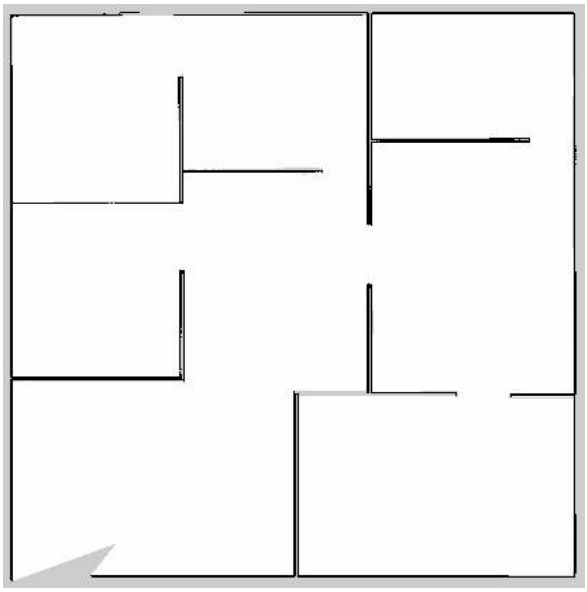


Figure 11: Final map obtained using Cartographer SLAM

Among the final 2D maps created using various SLAMs, the boundaries are well defined in the case of Hector SLAM. All other maps contain noises which may lead to poor navigation while integrating with navigation stack.

*B. Performance analysis of various SLAM*

The performance of the above mentioned SLAMs is evaluated based on the time taken to map the environment and reach the goal location. To achieve this, SLAM algorithm is integrated with Navigation stack, which makes the robot move autonomously to the goal location. The rqt graph integrating the SLAM algorithm and the Navigation stack is shown in Figure 12. The results from Table I show that hector SLAM is the most efficient among the four SLAM algorithms.

The results of both mapping and time taken to reach the goal location are better in the case of hector SLAM. Hence Hector SLAM is chosen in the proposed system for mapping.

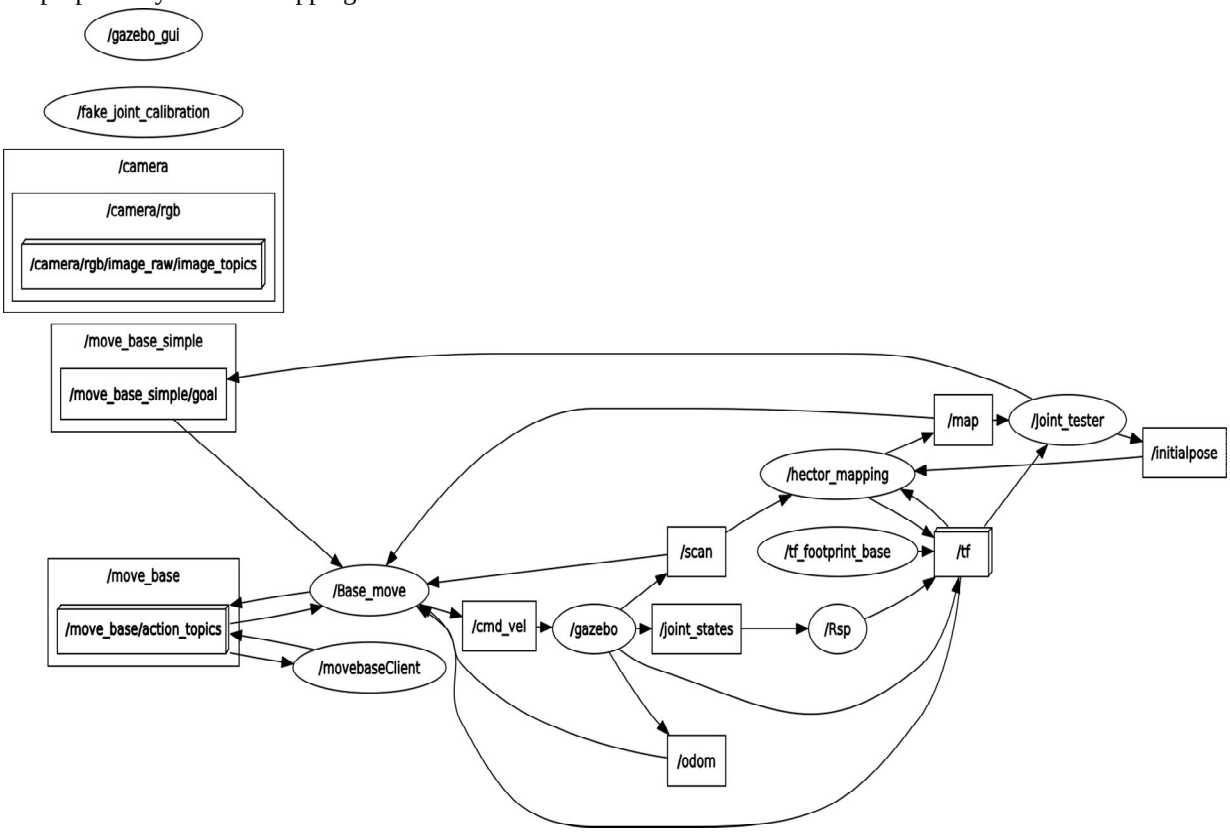



Figure 12: rqt graph integrating SLAM and Navigation stack

TABLE I
PERFORMANCE OF VARIOUS SLAM ALGORITHMS

| Sl No. | Gmapping (Time in seconds) | Hector SLAM (Time in seconds) | Karto SLAM (Time in seconds) | Cartographer (Time in seconds) |
|---|---|---|---|---|
| 1 | 105 | 85 | 94 | 88 |
| 2 | 122 | 105 | 123 | 126 |
| 3 | 110 | 125 | 92 | 118 |
| 4 | 131 | 122 | 94 | 92 |
| 5 | 116 | 90 | 110 | 117 |
| 6 | 112 | 92 | 95 | 125 |
| 7 | 90 | 78 | 100 | 118 |
| 8 | 102 | 92 | 112 | 124 |
| 9 | 108 | 82 | 116 | 103 |
| 10 | 90 | 98 | 88 | 93 |
| Average | 108.6 | 96.9 | 102.4 | 110.4 |

©IJRASET: All Rights are Reserved 

## IV. LOCALIZATION

A robot without correctly localizing itself in the constructed map cannot navigate and reach the goal location. For each step it takes, the position and orientation of the robot should be updated in order to take correct subsequent steps. So localization is an inevitable part in autonomous navigation robots.

### *A. Adaptive Monte Carlo Localization(AMCL)*

Adaptive Monte Carlo Localization method is used in this system to localize the robots in the given the map. AMCL is an algorithm based on particle filter method. A large number of particles are involved in this technique. At first, particles are sampled randomly. Each sample maintains a data regarding the position and orientation of robot. When robot takes a step, the particles are resampled based on their current pose and action taken by the robot using recursive Bayesian estimation.

The ROS package that implements this localization is amcl. Figure 13 shows the implementation of localization using AMCL algorithm in ROS for the proposed system. Green arrows around the robot represent particles, each carrying information regarding the position and orientation of the robot.

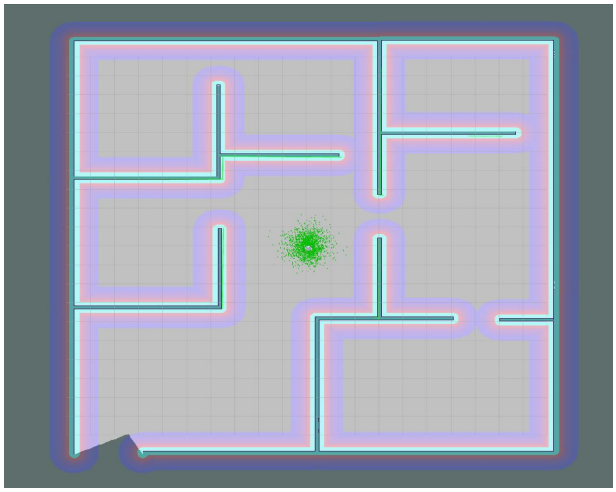


Figure 13: Adaptive Monte Carlo Localization (AMCL) implementation in ROS

amcl package takes map, lidar scan and odometry data as input as shown in Figure 14. After processing the input, it publishes “amcl pose” topic which provides the corrected odometry information.

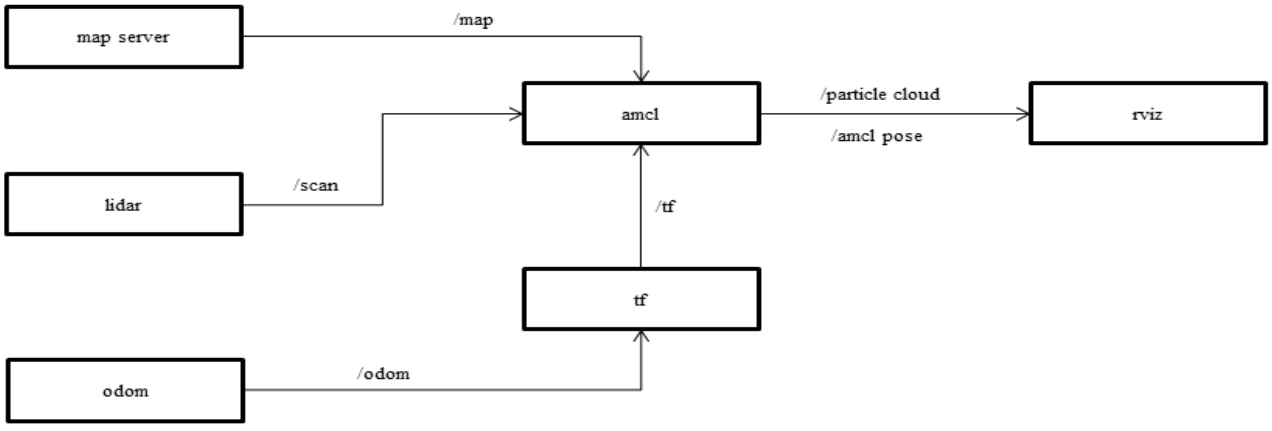



Figure 14: amcl package on ROS framework

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 10 Issue VI June 2022- Available at www.ijraset.com*

## V. NAVIGATION SYSTEM

For the autonomous navigation of robot, proper velocity commands have to be given to the robot based on goal location and robot pose. To implement autonomous navigation, Navigation stack in ROS is used in this system. Figure 15 shows the Navigation stack set up.

Sensor Transforms, Odometry source, SLAM and Goal Controller provides inputs to the Navigation stack. The inputs are processed by Navigation stack and it finally gives the velocity commands to the base controller. Lidar scan details are given by Sensor source package. Sensor Transforms package tells the physical position and orientation of different robot components such as robot base, wheel position, Lidar, camera etc. SLAM provides the map of the indoor environment which helps in planning the path of navigation of robot. The orientation and position of the robot is given by the Odometry source package. This helps to properly localize the robot. The goal controller provides the goal location to which the robot has to reach.

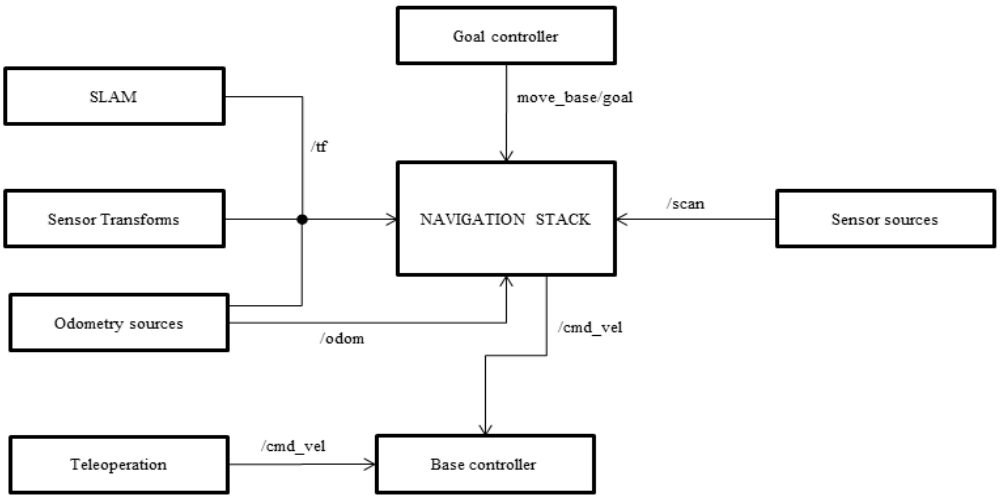



Figure 15: ROS Navigation stack setup

### *A. Differential Drive Model*

The proposed system uses a drive mechanism known as differential drive. A differential drive robot consists of two drive wheels which can be independently driven forward or backward. These two wheels are mounted on a common axis. To obtain the rolling motion, the velocity of the wheel has to be varied for each wheel. The point at which the robot rotates called Instantaneous Center of Curvature (ICC) lies on the common axis where the two wheels are mounted as shown in Figure16.

The relation between the linear velocity of motion of robot and the direction of movement of robot are:

*1)* If the velocity of right wheel and left wheel are equal, then the robot moves forward in a straight line.
*2)* If the velocity of right wheel and left wheel are equal, then the robot rotates about the midpoint of the wheel axis, i.e., the robot spins at stationary position.
*3)* If the velocity of right wheel and left wheel are not equal, the robot moves in the direction of wheel with lower velocity.

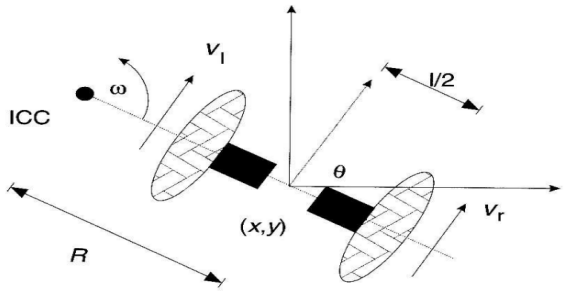



Figure 16: Differential Drive kinematics

©IJRASET: All Rights are Reserved 

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 10 Issue VI June 2022- Available at www.ijraset.com*

The velocity of right wheel $V_r$ and left velocity $V_l$ are given by equations 3 and 4 respectively.

$$V_l = \omega(R - \frac{l}{2}) \tag{3}$$

$$V_r = \omega(R + \frac{l}{2}) \tag{4}$$

where l is the distance of separation of the two wheels, R is the distance between ICC to the midpoint of the wheel and ω is the rate of rotation about the ICC. Equations 3 and 4 can be used to solve for R and ω at any instance of time. The expression for R and ω is by equation 5 and 6 respectively.

$$R = \frac{l}{2}\frac{V_r + V_l}{V_r \quad V_l} \tag{5}$$

$$\omega = \frac{V_r - V_l}{l} \tag{6}$$

*B. Odometry Model*

An Odometry source package provides the odometry information, i.e., location and orientation of the robot. If the robot is positioned at (x, y, θ) at time t, then the next position ($x^1$, $y^1$, $\theta^1$) can be calculated using equations 7 - 12.

$$x^1 = x + \delta x(t) \tag{7}$$

$$y^1 = y + \delta y(t) \tag{8}$$

$$\theta^1 = \theta + \delta\theta(t) \tag{9}$$

$$x^1 = x + \int_0^t V_s(t).cos[\theta(t)].\delta t \tag{10}$$

$$y^1 - y + \int_0^t V_s(t).sin[\theta(t)].\delta t \tag{11}$$

$$\theta^1 = \theta + \int_0^t \omega(t).\delta t \tag{12}$$

where, (x, y, θ) is the current coordinates of the robot, (δx(t), δy(t), δθ(t)) is the change in coordinates at time step t and $V_s(t)$ is the instantaneous velocity at the center point of the robot. The equation for $V_s(t)$ is given in equations 13 and 14.

To estimate the position and orientation for localization, this odometry model is required.

$$V_s = \frac{V_l + V_r}{2} \tag{13}$$

$$V_s - \omega R \tag{14}$$

©IJRASET: All Rights are Reserved 

*C. Transform Tree*

A transform provides details of position and orientation relationships between different components of robot attached to robot platform. In ROS, transforms forms a tree. This Transform (tf) tree contains nodes, each representing a frame. Each node has a parent node and many numbers of children. In a tf tree the location of a frame is specified with respect to its parent node. Figure 17 shows the tf Tree graph of the proposed system.

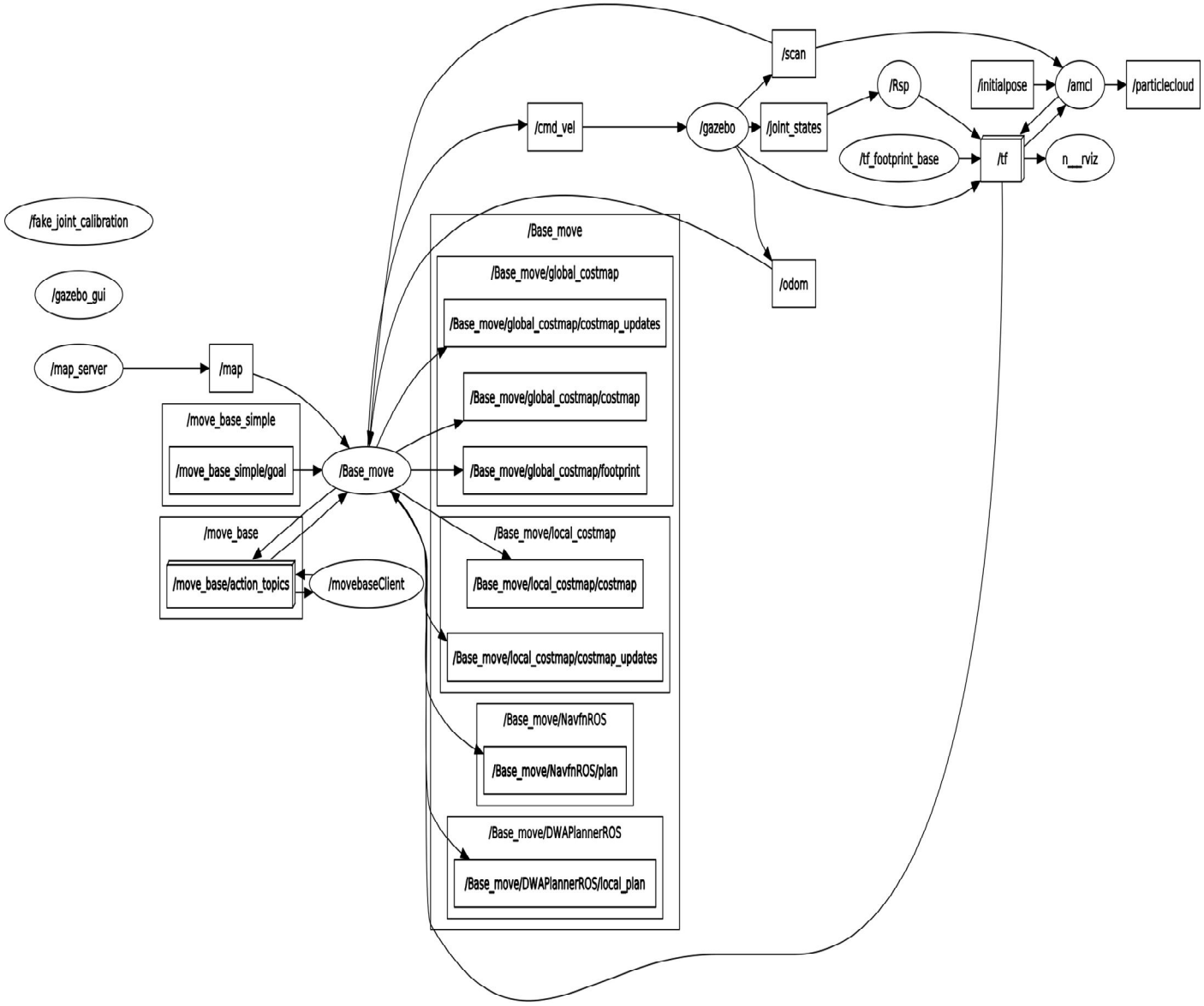



Figure 17: tf tree of the proposed system

*D. Tuning of Navigation Stack Parameters*

The fine tuning of parameters [8] is required for maximizing the performance Navigation stack. As the default parameters in the configuration file of ROS leads does not work for the navigation of the proposed system, the parameters are tuned for the system in such a way that it could accomplish the task of autonomous navigation without any fail at any instant. The tuned parameters used in this system are given in table II, III, IV and V.

TABLE II
LOCAL PLANNER PARAMETERS

| | |
|---|---|
| controller frequency | 10 |
| holonomic robot | false |
| yaw goal tolerance | 0.25 |
| xy goal tolerance | 0.15 |
| sim time | 0.4 |

TABLE III
COSTMAP COMMON PARAMETERS

| | |
|---|---|
| footprint | [[0.2,0.2],[0.2,-0.2],[-0.2,-0.2],[-0.2,0.2]] |
| transform tolerance | 0.5 |
| publish frequency | 10 |
| inflation radius | 1 |
| cost scaling factor | 3 |

TABLE IV
GLOBAL COSTMAP PARAMETERS

| | |
|---|---|
| update frequency | 10 |
| static map | true |

TABLE V
LOCAL COSTMAP PARAMETERS

| | |
|---|---|
| update frequency | 10 |
| static map | false |
| rolling window | true |
| Width | 3 |
| Height | 3 |
| Resolution | 0.05 |

## VI. PATH PLANNING

The task of indoor robot is to navigate in an indoor environment and reach a goal location. The performance of such robots can be maximized by choosing suitable path planning algorithm that finds an optimal path from the current position to the goal position. In the proposed system, A* algorithm is used as the global path planner. Global path planner fails to avoid a new or dynamic obstacle. This limitation is eliminated by incorporating the system with a local path planner called Dynamic Window Approach (DWA) [9].

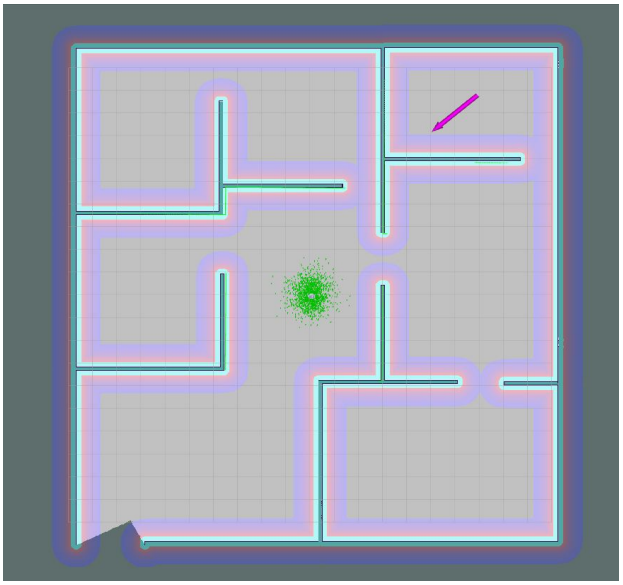


Figure 18: Goal location specified by the user

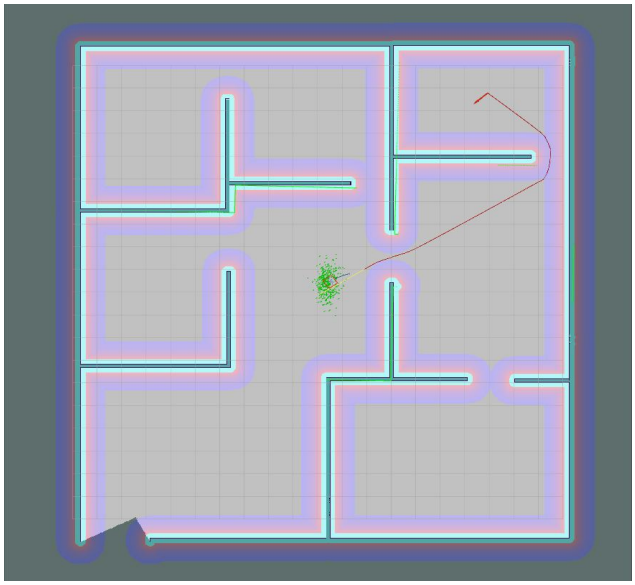


Figure 19: Optimal path obtained by using A* algorithm

### *A. A* Algorithm*

A* algorithm is a heuristic function based algorithm for finding optimal path from current location to target location. The A∗ algorithm uses the cost function to determine a path for the robot to move towards the end point. The cost function used in A* algorithm is given by equation 15.

$$f(n) = g(n) + h(n) \tag{15}$$

where f(n) is the estimated cost of the cheapest solution, g(n) is the cost to reach the current node (n) from the start state and h(n) is the cost to reach from node n to the goal node. In A* algorithm, h(n) is calculated using the Manhattan distance between the two points in space. The Manhattan distance between two points $(x_1, y_1)$ and $(x_2, y_2)$ is calculated using equation 16.

$$D_{Manhatten} - |x_1 - x_2| + |y_1 - y_2| \quad (16)$$

Figure 18 shows the goal location specified by the user. The optimal path obtained by using A* algorithm to reach the goal location is given in Figure 19.

*B. Dynamic Window Algorithm*

Local path planning algorithms are needed for autonomous navigation robots as this algorithms helps to avoid objects that are new or dynamic. In ROS, dwa_local_planner is used to implement Dynamic Window Approach (DWA) algorithm. This algorithm is based on the local costmap which provides obstacle information. At first, the DWA algorithm samples the velocity pairs in the velocity space. In the next step, for each velocity samples in robot's control space, the algorithm examines the circular trajectories resulting from forward simulation, and finally discards bad velocities. Based on the trajectory score, the algorithm then chooses the trajectory with the best score.

To evaluate the performance of DWA in the proposed system, an object is placed in the path of navigation of robot as shown in Figure 20. The Figure 21 shows how the robot plans a path to avoid the obstacle placed in its path of navigation.

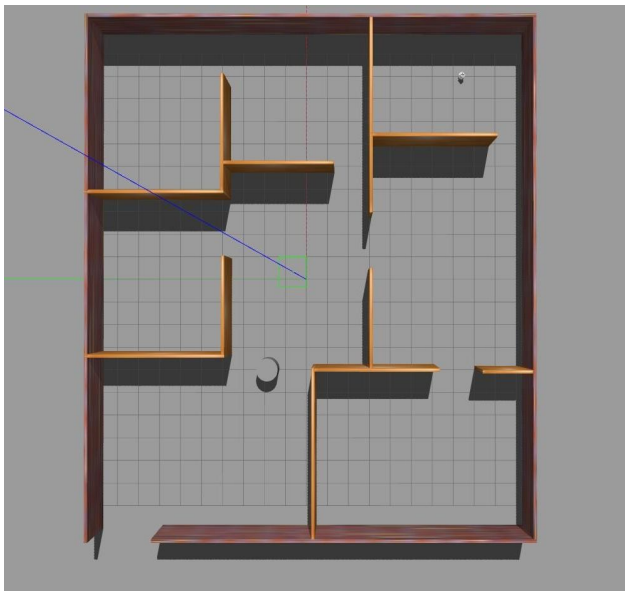


Figure 20: A new object placed in the simulated environment

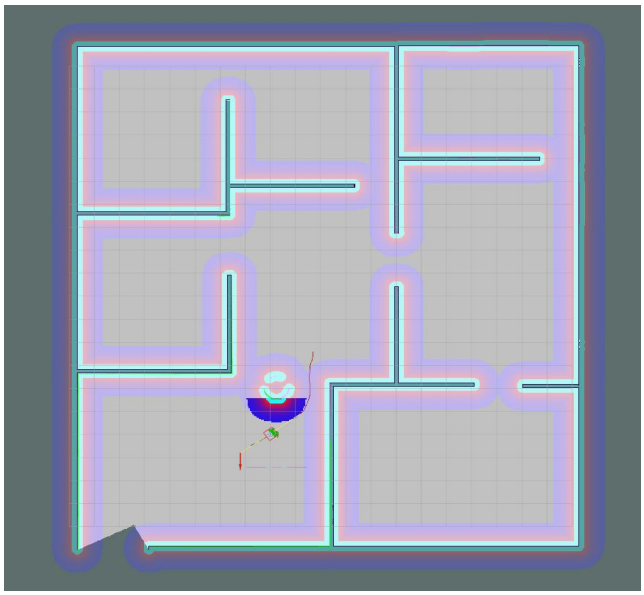


Figure 21: Path using DWA algorithm to avoid obstacle

## VII. OBJECT DETECTION

The performance of service robots can be improved by incorporating object detection capability with it. With object detection capability, robot is able to identify objects that come in the visibility of robot. Several deep learning models using Convolutional Neural Networks (CNNs) are available for object detection. These algorithms are mainly based on two approaches, i.e., one stage detector and two stage detector. In one stage detectors, the input image is divided into different regions and given to CNNs, resulting in detection of objects. On the other hand, in two stage detectors features are extracted from input image and region of interests (ROI). These two stage detectors are generally more accurate than single state classifiers. But they are slower because of multiple stages involved in the detection process.

YOLO is a single stage detector algorithm. This algorithm is popular due to its speed and accuracy. This paper studies different YOLO (You Only Look Once) versions [10]. YOLOv3 uses Darknet19 as backbone for object detection. But this model struggles in detecting small objects. YOLOv4, an improved version of YOLOv3 uses CSPDarknet53, helps in improving the speed and accuracy of the object detection algorithm. The latest version, YOLOv5 is the lightweight version of previous YOLO algorithms and uses PyTorch framework instead of Darknet framework. The comparison of these models is given in table VI. The detection of object by passing an image to these three YOLO versions pre-trained COCO is shown in Figures.

TABLE V
Comparison Of Different Yolo Versions

| Measure | YOLOv3 | YOLOv4 | YOLOv5l |
|---|---|---|---|
| Precision | 0.73 | 0.69 | 0.707 |
| Recall | 0.41 | 0.57 | 0.611 |
| F1 score | 0.53 | 0.63 | 0.655 |
| mAP | 0.46 | 0.607 | 0.673 |

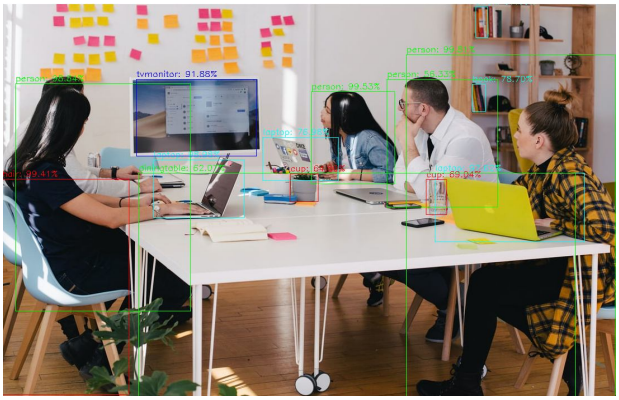


Figure 22: Detections in an image using YOLOv3



Figure 23: Detections in an image using YOLOv4

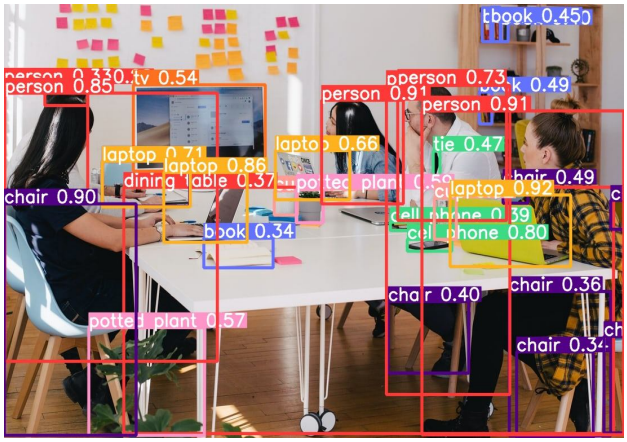


Figure 24: Detections in an image using YOLOv5l

The results show that number of objects detected of YOLOv5 is higher when compared with the other two versions.
At first, four models of YOLOv5 were available. The four models are Small, Medium, Large, and Extra-large. Now each model has two versions P5 and P6. P5 version has three output layers i.e., P3, P4, and P5. This version is trained on 640×640 images. P6 version has an extra output, p6 layer other than P3, P4, and P5.
As shown in Table VII, YOLOv5x of P5 model and YOLOv5l6 of P6 model has the highest mAP (Mean Accurate Precision) value. This system uses a model that combines YOLOv5x and YOLOv5l6 to obtain an improvement in performance of object detection. This technique of combining different models is generally known as ensemble technique.

TABLE VII
COMPARISON OF DIFFERENT YOLOv5 VERSIONS

| Measure | YOLOv5s | YOLOv5m | YOLOv5l | YOLOv5x | YOLOv5n6 | YOLOv5s6 | YOLOv5l6 |
|---|---|---|---|---|---|---|---|
| Size (pixels) | 640 | 640 | 640 | 640 | 1280 | 1280 | 1280 |
| mAP | 56.8 | 64.1 | 67.3 | 68.9 | 63.7 | 69.3 | 71.3 |

### *A. Performance evaluation of YOLO models*

A dataset consisting of 13 classes with 600 images is created for evaluating the performance of YOLOv5x, YOLOv5l6 models and ensemble model. The models are trained for 150 epochs. The mAP and recall values of these three models are shown in Table VIII.

TABLE VIII
PERFORMANCE EVALUATION

| Model | mAP | Recall value |
|---|---|---|
| YOLOv5x | 0.568 | 0.69 |
| YOLOv5l6 | 0.599 | 0.74 |
| Ensemble model | 0.607 | 0.76 |

The results show that there is an improvement in combining YOLOv5x and YOLOv5l6 models. So the proposed system used this model to detect the objects.

### *B. Integration of Ensemble Model with ROS*

The ensemble model is integrated with ROS for object detection. The real time images during the navigation of robot are taken by the camera placed in the robot and given as input to the ensemble model. This helps in detecting objects that are in the vicinity of robot. Figure 25 shows the detections using the developed ensemble model in a frame captured during the navigation of robot.

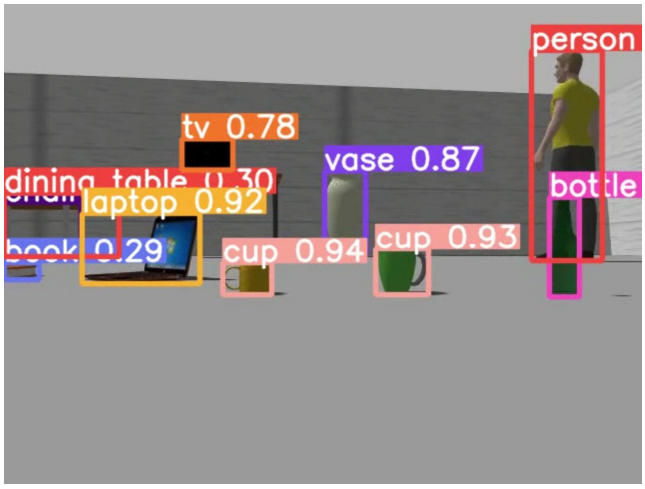



Figure 25: Detections in ROS environment using ensemble model

## VIII. CONCLUSION

This paper introduces an autonomous navigation robot for indoor services. The system integrates Hector SLAM, AMCL, A* and DWA algorithm. Also it is incorporated with an object detection algorithm that combines YOLOv5x and YOLOv5l6 algorithms. The proposed system was simulated in ROS. The simulation results proved that the proposed system succeeded in accomplishing the task of autonomous navigation of robot in an indoor environment. Also,tThe object detection model that combines the YOLOv5x and YOLOv5l6 algorithm is found have improvement the performance of detection. As a future work, the hardware of the proposed system can be made and evaluated. Also this sytsem can be extended to perform a specific indoor service robot.

## REFERENCES

[1] Xuexi Zhang, Jiajun Lai, Dongliang Xu, Huaijun Li and Minyue Fu, "2d Lidar-Based Slam and Path Planning for Indoor Rescue using Mobile Robots", Journal of Advanced Transportation, 2020.
[2] Y. Zhang, X. Wang, X. Wu, W. Zhang, M. Jiang and M. Al-Khassaweneh, "Intelligent Hotel ROS-based Service Robot," 2019 IEEE International Conference on Electro Information Technology (EIT), 2019, pp. 399-403, doi: 10.1109/EIT.2019.8834040.
[3] Zhang, Baoxian, Jun Liu and Haoyao Chen. "AMCL based map fusion for multi-robot SLAM with heterogeneous sensors" IEEE International Conference on Information and Automation (ICIA), 2013 , 822-827.

[4] P. Maolanon, K. Sukvichai, N. Chayopitak and A. Takahashi, "Indoor Room Identify and Mapping with Virtual based SLAM using Furnitures and Household Objects Relationship based on CNNs," 2019 10th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES), 2019, pp. 1-6, doi: 10.1109/ICTEmSys.2019.8695966.

[5] R. Raveendran, S. Ariram, A. Tikanmäki and J. Röning, "Development of task-oriented ROS-based Autonomous UGV with 3D Object Detection," 2020 IEEE International Conference on Real-time Computing and Robotics (RCAR), 2020, pp. 427-432, doi: 10.1109/RCAR49640.2020.9303034.

[6] P. Jensfelt, S. Ekvall, D. Kragic, and D. Aarno, “Integrating slam and object detection for service robot tasks,” in IROS 2005 Workshop on Mobile Manipulators: Basic Techniques, New Trends and Applications, Edmonton, Canada, 2005.

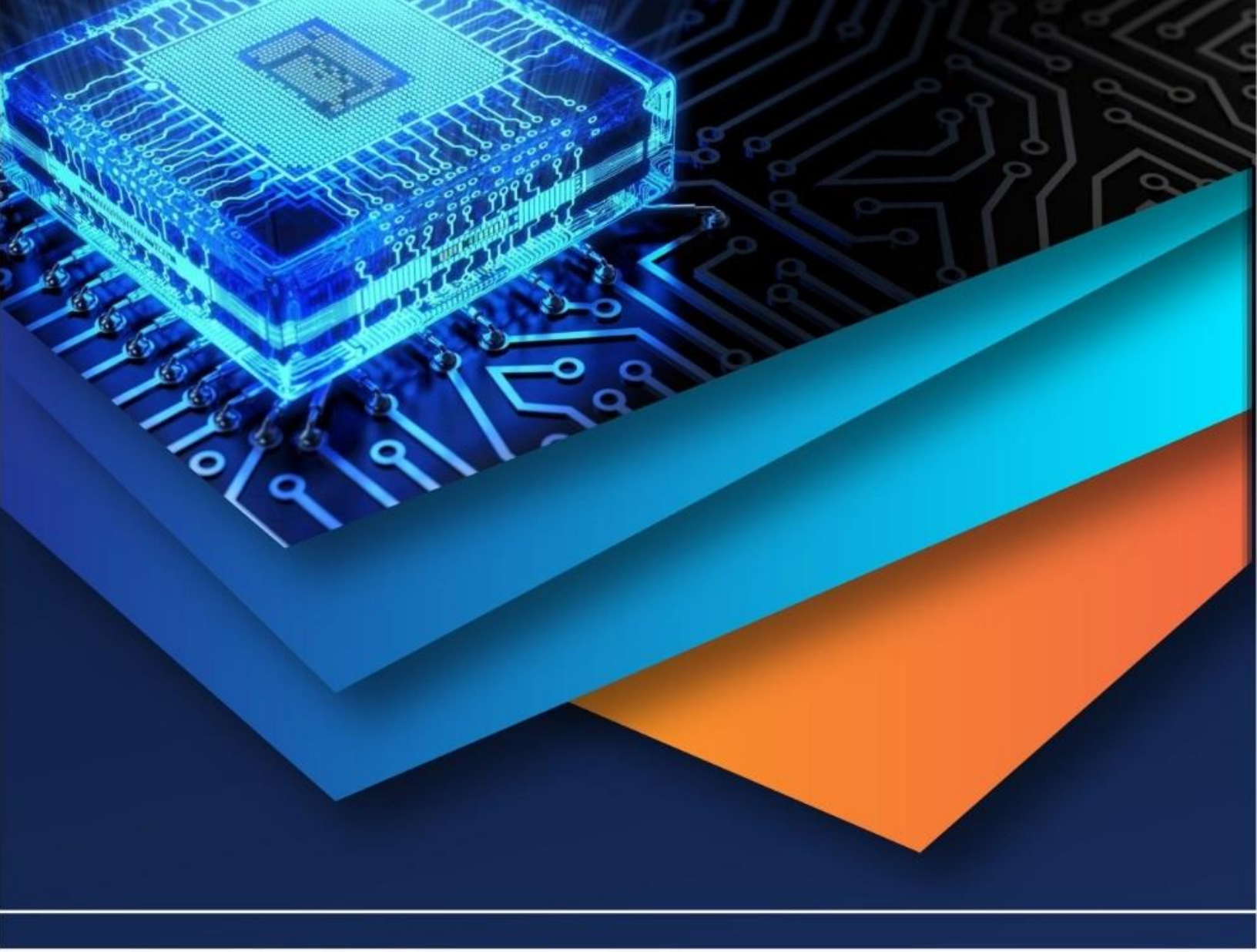
[7] M. Kulkarni, P. Junare, M. Deshmukh and P. P. Rege, "Visual SLAM Combined with Object Detection for Autonomous Indoor Navigation Using Kinect V2 and ROS," 2021 IEEE 6th International Conference on Computing, Communication and Automation (ICCCA), 2021, pp. 478-482, doi: 10.1109/ICCCA52192.2021.9666426.

[8] Zheng K," ROS Navigation Tuning Guide", 2017, Arxiv [Preprint],https://doi.org/10.48550/arXiv.1706.09068.

[9] Fox, Dieter & Burgard, Wolfram & Thrun, Sebastian, "The Dynamic Window Approach to Collision Avoidance", 1997 IEEE Robotics & Automation Magazine, Vol.4, pp. 23 - 33, doi:10.1109/100.580977.

[10] U.Nepal, H.Eslamiat, "Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty UAVs", 2022 Sensors 22, 464, https://doi.org/10.3390/s22020464.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)