# Industrial Warehouse Robot Simulation Using ROS

Pratik Padalkar[1], Pawan Kadam[2], Shantanu Mirajgave[3], and Aniket Mohite[4]

[1,2,3,4]Pimpri Chinchwad College of Engineering, Pune
padalkarpratik19@gmail.com[1]
pawankadam1107@gmail.com[2]
shantanu1058@gmail.com[3]
aniketmohite023@gmail.com[4]

**Abstract.** This paper presents the simulation of an industrial warehouse robot using ROS (Robot Operating System).In this paper industrial warehouse robot is successfully simulated. This robot can be used to carry the warehouse products from one point to another both manually as well as autonomously. To achieve the real-time autonomous feature SLAM(Simultaneous Localization and Mapping) is used to generate real-time environment maps so that robots can easily get localized in any kind of complex environment. Adaptive Monte Carlo Localization (AMCL) is used for the localization of the robot. Navigation stack is applied to the robot due to which the robot moves autonomously from one point to another.

**Keywords:** SLAM · TF · Gazebo · URDF · Navigation Stack · AMCL · Odometry · Gmapping · RViz.

## 1 Introduction

Robotics and Automation is the branch of science that deals with the process of building intelligent machines called robots to perform repetitive tasks. Earlier lots of research has been done to make robots autonomous. But in today's world as the robot demands are increasing exponentially to perform their daily tasks without any human interruption. So SLAM algorithms can be used to easily localize in any kind of environment so that robots can freely move without any kind of human interference. Industries need robots that can do work more efficiently, accurately, and without any human interaction. In industries, autonomous robots are the most demanding of all the robots.

The main objective of the robot is to self-explore around in the present environment and make smart decisions so that it can localize easily without any interface. The challenges faced in making robots autonomous are the environmental factors that contain numerous complex obstacles and unknown geographical landmarks. Another challenge is robot making, capable of navigating on its own without having any prior knowledge of the environment, trying to generate

its own map, making smart decisions based on collected data. The motivation behind this paper is to explore Robot Operating System and use of the tools provided to develop complex robots easily.

## 2 ROS - Robot Operating System

ROS is an abbreviation of Robot Operating System. It is an open-source robotics framework. It is a meta operating system, which works on almost all the operating systems like Linux, Windows, etc.[1]. The most commonly used ROS1 distributions are ROS-Indigo, ROS-Kinetic, ROS-Melodic, and ROS-Noetic. ROS has a wide range of community support across the world through its official community support and many other community groups are active in making ROS much stronger and more efficient.

ROS consists of *nodes, topics, clients, services, packages*, etc. through which it communicates with robots and its environment. ROS nodes are the basic working factors in ROS that perform computation[2]. ROS topic acts as a barrier that carries nodes and communicates between them. ROS services are a pair of messages, one for request and the other for replies, which can be sent over to different ROS topics. ROS client is a collection of code that makes writing different publishers and subscribers to the different ROS topics over which data can be passed. ROS package is the directory that contains all the necessary files like launch file, urdf file, params file, etc.

ROS can control multiple robots at the same time. ROS master can be connected to multiple robots at the same time due to which it is possible to control the robot simultaneously.

### 2.1 Gazebo

A gazebo is an open-source robotics simulation software. Gazebo simulates multiple robots in a 3D virtual environment. The gazebo is officially supported by ROS and is installed by default with certain ROS installations. It can be used for creating a virtual environment with obstacles in Gazebo which can be used with ROS for robot interface and configuration. Gazebo operates in two parts: the server which computes all the physics and the world and the client which is the graphical frontend for the gazebo.
To launch Gazebo, enter the following commands

```
$ roscore
$ rosrun gazebo_ros gazebo
```

### 2.2 RViz

RViz is an open-source visualization software. It gives us a convenient GUI to visualize our robot's exact position as well as surrounding environment maps. It allows us to view log sensor information from the robot's sensor. With the help

of RViz, we can visualize different types of sensor data such as camera, laser, etc. We can give a destination point to the robot through RViz.
To launch RViz enter the following commands on terminal

```
$ roscore
$ rosrun rviz rviz
```

### 2.3 Simultaneous Localization and Mapping (SLAM)

Simultaneous Localization and Mapping(SLAM) is a technique used in complex robots to generate a map around their present environment. There are various SLAM algorithms available. We can choose them according to our requirements. The map generated using this algorithm is then used by the navigation stack to move from one point to another autonomously[3]. The process of generating maps and localizing the robot is done concurrently where the maps are created dynamically by moving the robot in the present environment.

### 2.4 Uniform Robot Description Format (URDF)

URDF stands for Universal Robot Description Format. It is a description of a robot CAD model in ROS understandable format. It is an XML file describing the robot's physical attributes[4]. It is used to represent our robot and its physical attributes in simulations such as Gazebo and to visualize in RViz. URDF can be created by converting the CAD model into XML using CAD to the URDF exporter plugin[14].

### 2.5 Transforms (tf)

tf is a ROS package that is used to create multiple coordinate frames as well as track them over time[5]. tf can be viewed in RViz. It helps us to understand how different links of robots communicate with each other.

There are different frames in the robot which have relative motion between them. And the data between them is constantly changing as it is relative. So tf is used to convert the measurements from one frame to another.

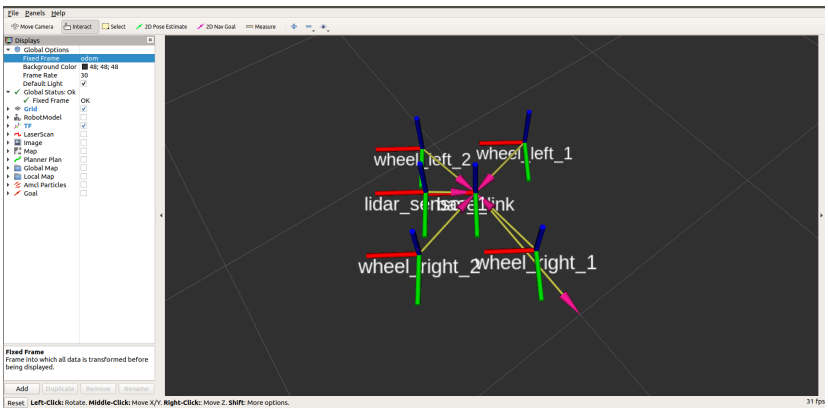


**Fig.1** tf diagram

### 2.6 Launch Files

Launch files are one of the basic file formats of the ROS. These files are of the format .launch and use a specific XML language format. These files provide users with a convenient way to interact with the ROS environment. Users can set up multiple ROS nodes in a single launch file and can also initialize and alter the various parameters according to the requirements.

## 3 Methodology

The industrial warehouse robot is designed and created from scratch and simulated in the ROS environment. To simulate a robot in the environment certain steps need to be followed. The steps are as follows:

### 3.1 Creating URDF from CAD Model

Fusion 360 is software that is used for 3D modeling, making CAD, CAM, CAE, etc. models. The first step for URDF generation is to design a robot model. There are some standard naming conventions to convert the CAD model into URDF[6]. The robot's base model should be given the name base_link which is important for exporting it to URDF. For exporting it to URDF the plugin URDF_EXPORTER is needed. This plugin not only generates urdf files but necessary launch files will be generated for spawning the robot in RViz and Gazebo. For moving the robot in Gazebo, a differential drive plugin needs to be added in URDF. After adding the plugin with the help of *joint_state_publisher_gui* we can simulate the robot in Gazebo and can visualize the same in RViz.

### 3.2 Creating a ROS Package

In the ROS package, the robot's URDF, all python and C++ script launch files, and also many other files that are required for making our robot autonomous are kept.
To create ROS package launch the below command

```
$ catkin_create_pkg iw_robot rospy roscpp amcl move_base
```

*iw_robot* is the name of our package and *rospy, roscpp, amcl, and move_base* are the other dependencies that we require for developing the robot.

### 3.3 Spawn in Gazebo

In order to spawn a robot in a designed environment for simulation purposes, we are using Gazebo. For spawning the robot first the launch files need to be created. As these launch files are coded in XML format, first the default programs need to be added to starting the gazebo, second and the most important thing is to add URDF files in it[7]. As URDF files are broken into multiple .xacro files so

main files are to be added. Then the controllers need to be added for controlling the robot. The controllers are usually written in a .yaml file. For spawning the robot in the gazebo, launch the following command. Refer Fig. 2

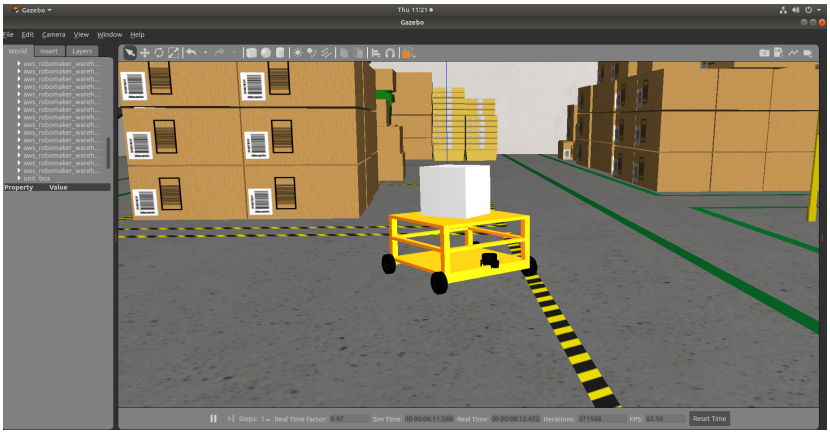

**Fig.2** Spawn in Gazebo

### 3.4 Spawn in RViz

For visualizing the robot in RViz necessary launch files need to be built. In this display. launch file and URDF file, RViz, and *robot_state_publisher* packages are added. For spawning the robot in RViz, launch the following command. Refer Fig. 3

```
$ roslaunch iw_bot display.launch
```

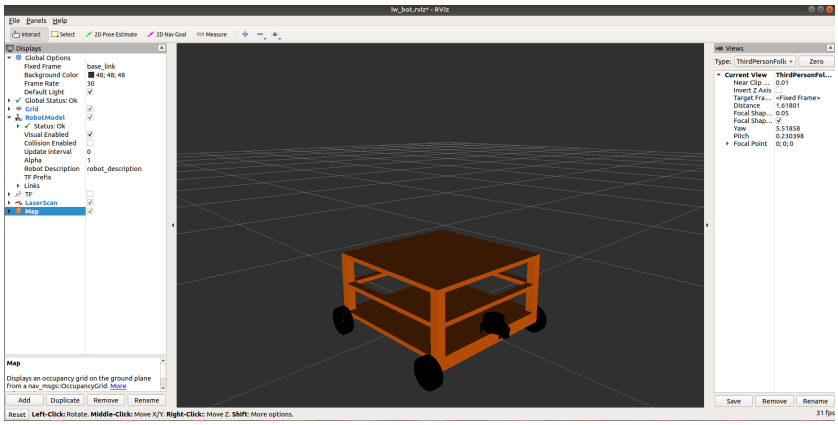

**Fig.3** Spawn in RViz

### 3.5 Teleoperating the Robot

To control the robot manually then we need to teleoperate it. Teleoperating means controlling the robot with the laptop's keyboard or with a joystick. To

teleoperate the robot, first launch the file having the name *iw_robot_teleop*. This launch file contains a *teleop_twist_package* containing a python script that publishes the robot's speed and the angle at which the robot has to travel. For teleoperating the robot there is a differential drive plugin available in iw_robot. URDF subscribes to our *cmd_vel* topic which is published by the *teleop_twist_package*[8]. To launch the *teleop_twist_package* run the following command:

```
$ rosrun iw_bot teleop.launch
```

### 3.6 Mapping

For creating a map of the environment Gmapping is used. Gmapping is a widely-used open source SLAM algorithm. It is a highly efficient Rao-Blackwellized particle filter-based algorithm which provides laser-based SLAM[9]. Gmapping has its official package to implement Gmapping called *slam_gmapping*. Gmapping acts as a ROS node and takes data from both laser sensor and robot pose and creates a 2D grid map of the environment[10]. This map can be visualized in RViz and retrieved via a ROS topic or service. To save this map we need the ROS map_server package which runs map_server and map_saver as a ROS node that reads the map and saves it to the local computer storage respectively. The map created is stored in a pair of files in the local computer storage. One is a YAML file (map_name.yaml) and the other is a pgm file (map_name.pgm). The YAML file contains the map meta-data and the image file name. Refer Fig. 4
To start the SLAM algorithm launch the following command:

```
$ roslaunch iw_bot gmapping.launch
```

In this launch file, all the necessary gmapping packages are added. This launch file will launch the Gmapping and RViz package and start creating the map in the present environment.

**Fig.4** Map created by Gmapping

### 3.7 Localization Of The Robot

Localization is a process in which a robot tries to locate with respect to its environment. To locate the robot in the environment it needs the robot's odometry which is provided by the gazebo itself. Localization transforms map frames to Odom frames. For localization of the robot, AMCL(Adaptive Monte Carlo Localization) algorithm is used[11]. The robot model uses a plugin of a laser sensor that spreads the particles in all possible directions. But for knowing the probable position of the robot on the map some filtering needs to be done. So it uses particle filters which filter the particle and determine the probable positions in the map. For using the AMCL algorithm, necessary launch files need to be created.
Command to launch the file.

```
$ roslaunch iw_robot amcl.launch
```

This launch file contains the various parameters that are required for this algorithm. These parameter files are configured according to our requirements.

### 3.8 ROS Navigation Stack

ROS Navigation stack is a ROS package used for navigating a robot autonomously from one location to another. It takes Odometry data, sensor data, and goal pose and gives velocity commands to the robot base. Refer Fig. 5 The Navigation Stack comprises of the following steps -

**Local Path Planning** Local Path Planning helps the robot to adapt its movement to a dynamic environment when obstacles are detected[12]. Local Path Planning in ROS is done by a local planner[13]. The local planner helps to move the robot in the environment. This planner calculates and publishes the robot's speed on a certain topic. Although of various local planners, DWA local planner if considered best as this planner gives more performance than any other planners. This algorithm performs forward simulation from the current state and predicts the path where the robot will not collide with the obstacles. This path planner is included in the *move_base.launch*. After launching this file the dwa local planner starts working[14].

**Global Path Planning** Global Path Planning is used to find the path between two points. For global path planning, ROS uses a global planner. This planner is included in the *move_base.launch*. After launching this launch file, the global planner starts to plan a path to reach its goal[13].

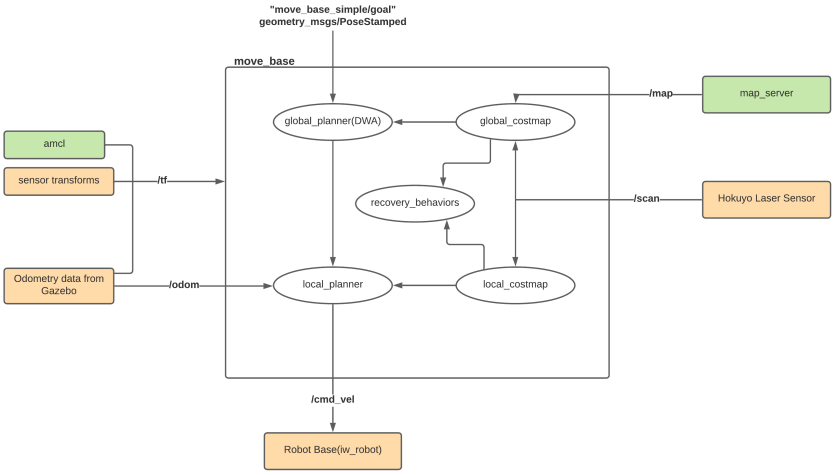


**Fig.5** Navigation Stack

**Navigation Goal** As all the necessary path planning algorithms are provided, it's time to move the robot to its destination. For moving the robot to any desired location there is a tool called 2D nav goal in RViz. Just we need to point at which position and orientation the robot should face on the map from its current position. Once given, the robot will start determining the best-desired path from the current position to the desired location with the avoidance of obstacles.
To start the navigation launch the given command

```
$ roslaunch iw_robot navigation.launch
```

All the launch files that we require such as *move_base,amcl* to run the navigation stack are included in this navigation.launch.

## 4 Conclusion

This paper describes the simulation of a ROS-based autonomous industrial warehouse robot. This robot can travel manually or autonomously which reduces human efforts. An industrial warehouse robot is spawned in the gazebo for testing purposes. This autonomous robot finds the shortest path which saves time and can be used as industrial robot to move any kind of goods autonomously from one certain place to another, without any human interaction and human effort. Thus, helping in increasing the automation in industrial areas.

Laser-based SLAM is analyzed and simulated on the robot. It uses the robot's Odometry data which is provided by Gazebo itself. By using the gmapping algorithm an incremental map of the environment is created. At last, the robot is localized in the map using AMCL, and a ROS navigation stack is used to implement the robot, due to which the robot moves autonomously from one point to another.

## References


1. Stefan Kohlbrecher and Oskar von Stryk ,Johannes Meyer and Uwe Klingauf.: A Flexible and Scalable SLAM System with Full 3D Motion Estimation. In: Nov 2011 IEEE. https://doi.org/10.1109/SSRR.2011.6106777
2. Robert Reid, Andrew Cann, Calum Meiklejohn, Liam Poli, Adrian Boeing, Thomas Braunl.: Cooperative multi-robot navigation, exploration, mapping and object detection with ROS. In: October 2013 IEEE. https://doi.org/10.1109/IVS.2013.6629610
3. Werede Gunaza Teame, Dr.Yanan Yu, Wang Zhongmin.: Optimization of SLAM Gmapping based on Simulation. In: April 2020 IJERT International Journal Of Engineering Research Technology. https://doi.org/10.17577/IJERTV9IS040107
4. Maria Gjerde Lill,Johannessen,Mathias Hauan Arbo.: Robot Dynamics with URDF CasADi. In: Nov 2019 IEEE. https://doi.org/10.1109/ICCMA46720.2019.8988702
5. Tully Foote: Tf: The Transform Library. In:July 2013 IEEE. https://doi.org/10.1109/TePRA.2013.6556373
6. Toshinori Kitamura, Fusion2URDF,https://github.com/syuntoku14/fusion2urdf. Last accessed 2021/10/24
7. Ilya Shimchik, Artur Sagitov, Ilya Afanasyev, Fumitoshi Matsuno and Evgeni Magid.: Golf cart prototype development and navigation simulation using ROS and Gazebo.In: May 2016 ResearchGate.
8. Donghyeon Lee, Young Soo Park.: Implementation of Augmented Teleoperation System Based on Robot Operating System (ROS). In: Oct 2018 IEEE. https://doi.org/10.1109/IROS.2018.8594482
9. Zhang Xuexi, Lu Guokun, Fu Genping, Xu Dongliang, Liang Shiliu.: SLAM Algorithm Analysis of Mobile Robot Based on Lidar. In:October 2019 IEEE. https://doi.org/10.23919/ChiCC.2019.8866200
10. Kirill KrinKin, Artyom Filatov, Artur Huletski, Dmitriy Kartashov.: Evaluation of Modern Laser Based Indoor SLAM Algorithms. In:May 2018 ResearchGate. https://doi.org/10.23919/FRUCT.2018.8468263
11. Wallace Pereira Neves dos Reis, Guilherme Jose da Silva, Orides Morandin Junior, Kelen Cristiane Teixeira Vivaldini.: An Extended Analysis on Tuning the Parameters of Adaptive Monte Carlo Localization ROS Package in an Automated Guided Vehicle. In:Feb 2021 ResearchGate. https://doi.org/10.21203/rs.3.rs-225880/v1
12. Rasika Kangutkar, Jacob Lauzon, Alexander Synesael, Nicholas Jenis, Kruthika Simha, Raymond Ptucha.: ROS Navigation Stack For Smart Indoor Agents. In:Sept 2018 IEEE . https://doi.org/10.1109/AIPR.2017.8457966
13. Pablo Marin-Plaza, Ahmed Hussein, David Martin and Arturo de la Escalera.: Global and Local Path Planning Study in a ROS-Based Research Platform for Autonomous Vehicles. In:Feb 2018 ResearchGate. https://doi.org/10.1155/2018/6392697
14. Xuexi Zhang, Jiajun Lai, Dongliang Xu, Huaijun Li and Minyue Fu.: 2D Lidar-Based SLAM and Path Planning for Indoor Rescue Using Mobile Robots. In:Nov 2020 Hindawi. https://doi.org/10.1155/2020/8867937