RoboAssist: Simulation of an ROS based Robot for Assisting Old and Differently abled

Ahzam Zobairi  
B.Tech. Student,  
Dept of ECE, Lords Institute of Engineering & Technology,  
Hyderabad, India  
e-mail: ahzam.zobairi96@gmail.com

Mohammed Shaji  
B.Tech. Student,  
Dept of ECE, Lords Institute of Engineering & Technology,  
Hyderabad, India  
e-mail: mohdshaji707@gmail.com

Abdul Wasay Mudasser  
Associate Professor,  
Dept of ECE, Lords Institute of Engineering & Technology,  
Hyderabad, India  
e-mail: abdul.wasay@lords.ac.in

Abstract—This paper presents an intelligent 7-DOF Android Robot to be used for assisting old and differently abled in homes and in their day to day lives. The robot utilizes a special OpenCV script that detects the host and registers them as master. The robot then continuously monitors the registered master and serves them as required. Some motions such as falling and fainting has been pre-recorded which when detected initializes the Emergency Service of the Robot. The open-source meta operating system ROS (Robot Operating System) has been used to develop the framework. The designed robot is simulated in Gazebo.

Keywords—ROS, DOF, Robot, Gazebo, MoveIt, SLAM, OpenCV, Rviz, rQt, Open-Source

INTRODUCTION

Due to the advancement in technology and the highly competitive and globalized world of today, a large number of people are left neglected. Since ancient times the elderly and the differently abled were treated with care by their family members. But that’s not the case of today’s world. People blinded by greed of growth and self-satisfaction neglect them as they opt a self-centred mentality. To assist these neglected people in their homes and day to day lives, a robot based on ROS has been developed.

ROS stands for Robotic Operating System [1]. It is an open-source meta operating system commonly used by researchers and hobbyists to design and develop their own robot. ROS by itself is just a service which helps in communication between large number of devices within a system. However, it becomes a very powerful tool due to the large number of powerful packages and libraries. Some commonly used packages include: MoveIt, Rviz, Gazebo and SLAM. ROS is expected to be soon the industry and academia standard for Robotics. Due to its wide of packages the prototyping of experimental hardware and software with the ease in complexity leads to exponential growth [2].

In this paper, a robot in the model of an android has been designed for assisting the aforementioned. The robot is built on a number of ROS packages. However, what makes this robot special is its ability to continuously monitor its master and to take actions as instructed and/or as needed. A script using OpenCV4 is being used to do so.

This paper is divided in two sections. Section I describes about the model of designed robot. Section II discusses about the various sensor outputs and general simulation of the robot in Gazebo.

SECTION I: ROBOT 3D DESIGN

The model of the robot is inspired by the logo of Android operating system as the body will have the major electronics hardware component and the transparent one-way hollow dome that will have the component for robot to perceive the information about the environment. Instead of bipedal we have replaced it by short cylinder that acts as
omnidirectional platform helping it to move around. The robot model has been designed using SolidWorks and later imported into the URDF file for use in Rviz and SDF used by Gazebo.

The above screenshot displays the 3D model of the designed robot in SolidWorks. As you can observe, at its base is cylindrical omnidirectional platform that have four mechanism wheel which can be controlled individually by motor control IC’s. For easy control and navigation around any surface mecanum wheel is used.

The mid part of the robot consists most of the hardware used as described in Table-X. A 6-DOF arm is attached to its side for operations like pick and place. Together with the base’s linear motion, the robot has 7 DOF manipulation.

SECTION II: SIMULATION

The robot is simulated by using Gazebo, a powerful simulator powered by OGRE-3D engine used extensively in the research and development of robot across the world.

Although the robot is designed in SolidWorks its file format is not supported by ROS Tools and Packages and had to be converted into URDF (Unified Robot Descriptive File) format. The URDF file is directly accepted by Rviz, however for Gazebo we had to modify our URDF file by adding collision and other simulation specific properties before converting it into SDF (Simulation Descriptive File).

SECTION II

A. MoveIt

MoveIt employs ample plugin architecture and object motion planning that can be easily be integrated with major packages [3]. System architecture is very high up for primary node and made available by MoveIt called move_it group. For implementation we need

User interface: - It is done either by any of three ways in C++, in python and through GUI.
Configuration :- move_ros is a ROS a node and required 3 information URDF, SRDF, MOVE IT configuration.
Robot interface :- The communication with robot to get its current stance and make a point cloud and to communicate to the controller of the robots.

Motion planning :- It has three steps
The motion planning plugin :- Motion planning with MoveIt done by plugin interface that’s allow MoveIt to interact and employ other plugin that extends it. For default OMPL is configured as motion planner and MoveIt setup assistant interfaces MoveIt and OMPL.
Motion plan request :- It clearly describe what the motion planner has to do or you like it to do like moving its manipulator. Collisions are checked for by default including self-collision. A gripper as an end effector can be interfaced.
Motion plan result :- Responding motion plant request move it creates the trajectory and this trajectory will help to move any movable part to a desired location.
B. Simultaneous Localisation and Mapping

As of robot needs to work in arbitrary environment, we need to localise the robot and map the environment. It can be achieved using localisation and mapping. That generally two separate methods for localisation and mapping with different algorithms that are

Localisation :- It estimates the pose of the robot provided with the map of the environment and access to it sensory data. Robot takes few trials moving around before narrowing it down to precise pose as of X and y co-ordinate along with its orientation.

Mapping :- It uses the known pose and access to movement along with the sensory data to produce the map of environment using trajectory and sensors. A pose is defined by finite number of x and y co-ordinate with orientation but map is a continuous that means infinite number of X and y variable present. And uncertainty with perception is present that will lead to more challenges along with workspace and geometry of objects. The algorithm that we used is

Occupancy Grid Mapping Algorithm :- Mapping of arbitrary environment by dividing into finite number of grid cell and estimating each cell. Estimating each cell, we will estimate the whole environment.

SLAM abbreviated as Simultaneous Localisation And Mapping that estimate pose of the robot and mapping of the environment simultaneously[4]. It creates map simultaneously while localising itself with respect to the map and it is more challenging than localisation and mapping because we do neither have map for localisation and nor pose needed for mapping. And there will definitely have noise in robotic motion and measurement of maps along with uncertain pose that will take up to error in pose and map which are correlated. The accuracy of map is directly related with the accuracy of localisation of robot. So here we come towards the chicken and egg problem, as map is needed for localisation and robot pose is needed for mapping. That makes Slam challenge and fundamental to mobile robotics because the robot must know how to move in arbitrary environment as it is not feasible to provide the map of every object in the vicinity because the movement of object can be dynamic.

We are using SLAM, so no pose and maps are available. Our robot uses set of different perceiving hardware like 3D Sensor along with laser sensor to localize its pose and mapping the environment [5]. The laser sensor is used so that the object and its distance with respect to the position of the robot is determined along with its pose. As we will get the pose, we will use 3D sensor to Navigate around and collecting the data to build a environment around it with the provided sensory data and access to movement. This makes our robot to fit for any arbitrary environment.

C. OpenCV

OpenCV is huge open source library with various of integration on it like computer vision, image processing and machine learning especially with deep learning [6]. Now OpenCV making a huge difference in technology as it is employed in real time on system as we have used.

The integration of computer vision, deep learning and image processing directly related to object detection that will detect object instances from the live feeds. The technique used is below

Haar Cascade :- It basically uses the concept of positive and negative images to train the classifier as machine learning based approach.

Positive Images :- The images that needed to be identified by classifiers.

Negative Images :- Everything else in the vicinity excluding the positive image.
By the means of OpenCV deployed on the 3D sensor we can be able to use it with the integration of Laser sensor for both the object detection, collision as well as the navigation through the collision.

![Fig-2.3 Object detection using OpenCV](image1)

**D. Gazebo**

ROS integrates closely with Gazebo through the gazebo_ros package[7]. This package provides a Gazebo plugin module that allows bidirectional communication between Gazebo and ROS. Simulated sensor and physics data can stream from Gazebo to ROS, and actuator commands can stream from ROS back to Gazebo. In fact, by choosing consistent names and data types for these data streams, it is possible for Gazebo to exactly match the ROS API of a robot. When this is achieved, all of the robot software above the device-driver level can be run identically both on the real robot, and (after parameter tuning) in the simulator. Lastly it is an enormously powerful tool for simulation.

![Fig-2.4 simulation in gazebo](image2)

**CONCLUSION**

This paper proposed a method to control the manipulator automatically to fetch the object and provide it to master. The simulation helps us to know the robot parameters like orientation, navigation, object recognition, and position. The design of the robot is done using CAD, the robotic visualization done using RVIZ and simulation is done using GAZEBO. The simulation in gazebo is used to explore more realistic situations. The testing is done and results are successfully acquired.

**REFERENCES**


