IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 11, Issue 5, May 2023

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.379

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 |

|| Volume 11, Issue 5, May 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1105148 |

Three-Wheeled Common Belt Swerve Drive Autonomous Navigation using ROS

Shraddha Gorde, Avdhoot Ubale, Jainam Jain, Mukta Jamage

UG Student, Dept. of I.T., Pimpri Chinchwad College of Engineering, Pune, India

UG Student, Dept. of I.T., Pimpri Chinchwad College of Engineering, Pune, India

UG Student, Dept. of I.T., Pimpri Chinchwad College of Engineering, Pune, India

Assistant Professor, Dept. of I.T, Pimpri Chinchwad College of Engineering, Pune, India

ABSTRACT: This research paper presents a novel approach to autonomous navigation using a three-wheeled common belt swerve drive system controlled through the Robot Operating System (ROS). The integration of ROS provides a flexible and efficient platform for developing autonomous navigation algorithms and facilitates seamless communication between the robot's hardware and software components. The proposed three-wheeled common belt swerve drive system offers enhanced maneuverability and stability, enabling the robot to navigate through complex environments with ease. The system utilizes a belt-driven mechanism, which allows dependent control of each wheel's direction. The navigation algorithms implemented within the ROS framework leverage the sensor data to perform tasks such as mapping, localization, and path planning. The robot builds a map of its environment using simultaneous localization and mapping (SLAM) techniques, allowing it to navigate efficiently and avoid obstacles. Path planning algorithms generate optimal paths based on the map and the robot's destination, ensuring smooth and collision-free navigation. This research paper contributes to the field of autonomous robotics by presenting a comprehensive framework for three-wheeled robot navigation using a common belt swerve drive system and ROS. The combination of the mechanical design, sensor integration, and ROS-based navigation algorithms offers promising possibilities for various applications, including service robots, indoor mapping, and surveillance systems.

KEYWORDS: Robot Operating System, Simultaneous Localization and Mapping, Adaptive Monte Carlo Localization, Unified Robotics Description Format, Autonomous Mobile Robot, Autonomous Guided Vehicle

I. INTRODUCTION

Autonomous navigation, a fundamental aspect of autonomous robotics, involves enabling robots to traverse complex environments independently, making decisions and avoiding obstacles in real-time. Such capabilities are essential for applications ranging from service robots in homes and healthcare facilities to exploration robots in challenging terrains. In this research paper, we focus on the development of a three-wheeled common belt swerve drive system for autonomous navigation using the Robot Operating System (ROS). The choice of a three-wheeled configuration offers several advantages over traditional four-wheeled systems, including increased maneuverability and stability. By utilizing a common belt swerve drive mechanism, we enable precise and agile movements, allowing the robot to navigate through narrow spaces and negotiate obstacles effectively. The integration of ROS as the underlying framework for our autonomous navigation system brings numerous benefits. ROS provides a modular and extensible platform for developing robotic applications, facilitating the seamless integration of hardware components, sensors, and algorithms. It offers a standardized communication interface, allowing different modules to exchange data efficiently. Additionally, ROS provides a vast collection of libraries and tools, simplifying the development and deployment of autonomous navigation algorithms.

To enable autonomous navigation, our system incorporates a suite of sensors, including laser range finders, cameras, and IMUs. These sensors provide the robot with rich environmental data, enabling it to perceive its surroundings and make informed decisions. Laser range finders generate accurate distance measurements, while cameras capture odometry information. IMUs provide information about the robot's orientation. The data from these sensors are processed and fused to create a comprehensive representation of the robot's environment and movement.

Within the ROS framework, our autonomous navigation system employs a range of algorithms to perform essential tasks such as mapping, localization, and path planning. Simultaneous Localization and Mapping (SLAM) algorithms enable the robot to build a map of its environment in real-time, incorporating both static and dynamic obstacles. Localization algorithms use the map and sensor data to estimate the robot's position and orientation accurately. Path

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 |

|| Volume 11, Issue 5, May 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1105148 |

planning algorithms generate optimal paths based on the map and the robot's destination, considering factors such as obstacle avoidance and smooth trajectory planning. The contributions of this research lie in the development of a novel three-wheeled common belt swerve drive system for autonomous navigation and its integration within the ROS framework. The combination of mechanical design, sensor integration, and ROS-based navigation algorithms offers promising possibilities for a wide range of applications in the field of autonomous robotics. The findings presented in this research paper serve as a foundation for further advancements in autonomous navigation systems, enabling the deployment of intelligent and capable robots in various domains.

II. RELATED WORK

In [1] the authors aimed to design an indoor Autonomous Mobile Robot (AMR) that possesses similar capabilities to commercial AMRs currently used in industries. To achieve this, they utilized the Robot Operating System (ROS) and implemented a navigation package program provided by ROS. The authors developed specific packages for the robot, such as the odometry node, IMU node, and robot transformation node, to enable communication and message exchange between the ROS nodes and the robot's devices.

In [2] the authors focused on designing an Autonomous Mobile Robot (AMR) using the Robot Operating System (ROS) and incorporating lidar as the sensing sensor. They also developed a custom wire-controlled chassis to serve as the mobile platform for the AMR. The AMR employed the Cartographer algorithm, a mapping algorithm within ROS, along with the Move Base path planning algorithm. This combination allowed the AMR to efficiently track and navigate its surrounding environment, as well as reach target points effectively.

In [3] the authors of the research paper designed an AMR adapted for ROS by considering both the hardware architecture and electronic communication protocols. They obtained the necessary components, mounted them on the platform, installed interconnections and software, conducted tests, and proposed solutions for encountered problems.

In [4] the authors introduce a novel mapping strategy to build a hybrid grid-topological-semantic map, which serves as the basis for voice-control-based navigation for the robots. This mapping strategy allows the robot to navigate based on voice commands in a structured environment. The authors conducted experiments to demonstrate the effectiveness of their proposed mapping strategy in combination with the voice recognition system for achieving voice-controlled navigation. The results indicate that the mapping strategy and voice recognition system work well together, enabling accurate navigation based on voice commands.

In [5] The mobile robot is controlled using an Arduino microcontroller and establishes communication with the cloud through a Raspberry Pi. To facilitate the cloud infrastructure, the authors set up a private cloud using OpenStack, which provides Infrastructure as a Service (IaaS). This cloud infrastructure allows for efficient storage and retrieval of the collected data. he data collected by the robot are stored in a cloud server, which can be accessed and viewed through a web browser. This data can be analyzed to create awareness and understanding of environmental changes in the location under study.

In [6] the authors aimed to leverage the capabilities of ROS, such as data analysis tools, support for multiple robots and sensors, and teleoperation device interaction, to enhance engineering education. The authors conducted a variety of experimental tests using the algorithms developed for the virtual agents and implemented them on the physical robotic platforms. These tests aimed to validate the effectiveness and performance of the algorithms in real-world scenarios.

III. PROPOSED ALGORITHM

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 |

|| Volume 11, Issue 5, May 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1105148 |

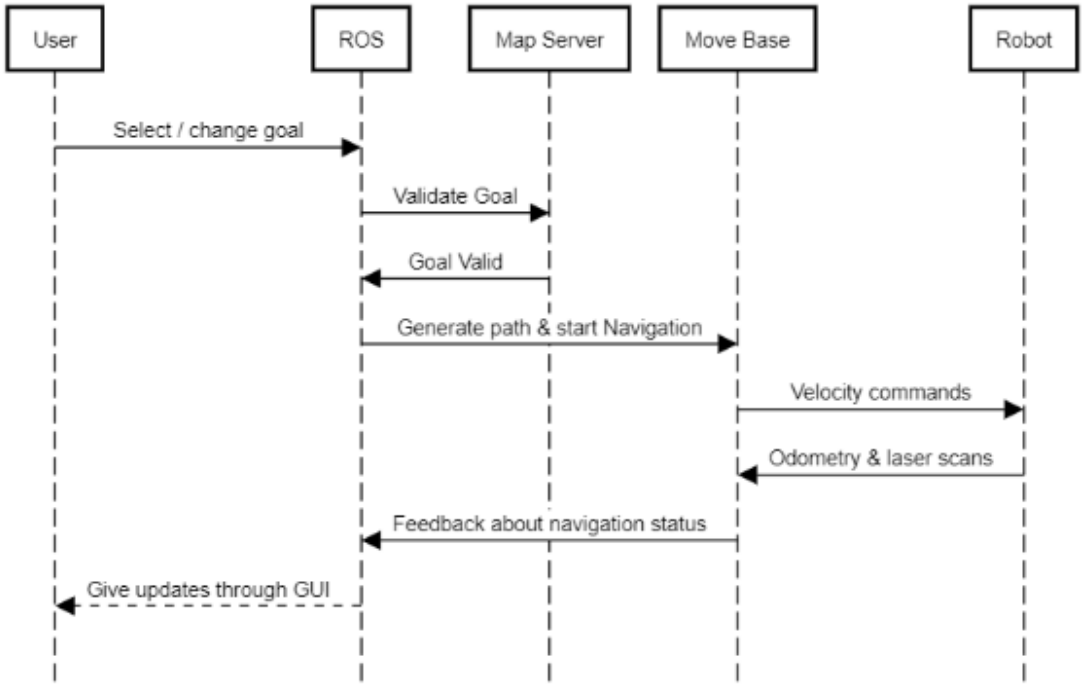

Figure 1: Sequence Diagram

This architecture has a ROS server running on a desktop or in a microprocessor like Jetsonxavier or Jetson Nano. The processor is connected with a microcontroller which is on the robot connected with all the actuators on a PCB. The robot has two more important components on it. First one is the odometry source, in our case it is an Intel Realsense T265 tracking camera which uses OpenCV to detect motion and calculate velocities, angles and positions. The second one is a laser distance sensor, in our case it is RPLidar A1 sensor.

There will be two nodes running on the processor one for odometry and one for laser scans. Both nodes will be getting raw data from odometry source and laser distance sensor respectively and will publish that data on respective topics. Few other important nodes are also running on the processor those are - map_server, move_base, amcl, etc. These nodes use most of the processing power of the processor.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 |

|| Volume 11, Issue 5, May 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1105148 |

```
<launch>
 <include file="$(find fwomni_hardware)/launch/display.launch" />
 <arg name="map_file" default="$(find fwomni_hardware)/maps/ring_arena.yaml"/>
 <arg name="move_forward_only" default="false"/>

 <node pkg="map_server" name="map_server" type="map_server" args="$(arg map_file)"/>

 <node pkg="amcl" type="amcl" name="amcl">
        <remap from="scan" to="$(arg scan)"/>
        <param name="odom_frame_id" value="camera_odom_frame"/>
        <param name="base_frame_id" value="base_link"/>
 </node>

 <node pkg="move_base" type="move_base" respawn="false" name="move_base" output="screen">
        <param name="base_local_planner" value="dwa_local_planner/DWAPlannerROS" />
        <rosparam file="$(find fwomni_hardware)/params/dwa_local_planner_params.yaml" command="load" />
        <remap from="cmd_vel" to="$(arg cmd_vel)"/>
        <remap from="odom" to="$(arg odom)"/>
 </node>
</launch>
```

Figure 2: The ROS nodes launch file

A GUI is available for interaction of the user with the system. From there the user can send goals to the user, the goal first goes at map_servernode. Map server node checks the reachability of the goal position. If the goal is reachable then map_server gives that goal to move_base node. That node then creates global and local plans to reach the goal. If the goal is not reachable then the map_server node notifies us through CLI. The amcl node parallelly takes input from laser scans and transforms of laser distance sensor to base_link and creates transforms from world frame to odom frame. The other transforms below the base_link comes from URDF by robot_state_publisher. The transforms between odom and base_link needs to be created by the node which publishes the odometry data.

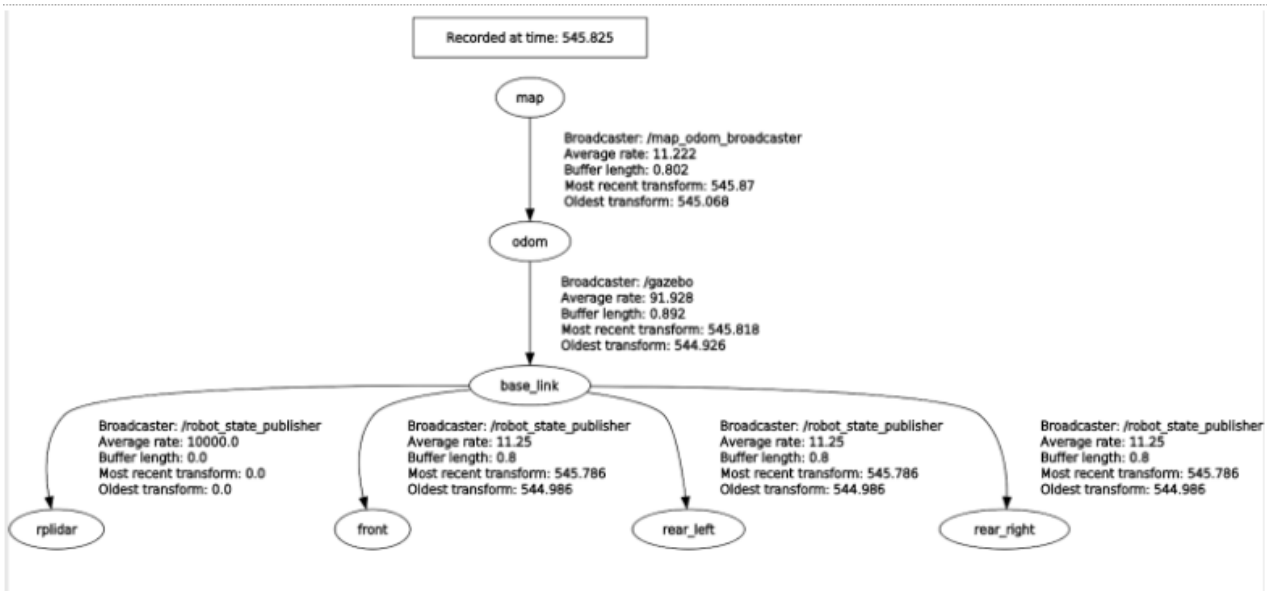

Figure 3: TF Tree

In above tf tree, we have created our own map to odomtf broadcaster, we didn't let amcl to publish transforms due to processing power limitations of our processor. The other reason for doing the same is our world origin and map origin are the same, so there was no point in creating localization based map to odom transforms.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 |

|| Volume 11, Issue 5, May 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1105148 |

This all were the algorithm or tasks getting executed on the processor, now let's talk about the code running in the microcontroller. Our microcontroller is an Arduino Mega. Its buffer size is 256 bytes for a single message through serial communication. The arduino and ROS are connected using the ROS topic. Here all communication is serial communication. ROS by default uses serial0 of the arduino so we can't use the serial monitor while the communication is going on.
The code running on arduino mega is the inverse kinematics of three wheeled swerve drive and the code for actuating all the actuators of the robot. Our robot is a common belt driven three wheeled swerve drive so all the wheels are always aligned to a single direction. For this code, the input is the net velocity of the robot and the output of the inverse kinematics equation is the angle of the wheels and velocity of the wheel. Let's suppose x and y are the velocities in m/s of the robot in 2D plane published by move_base. So according the inverse kinematics equation -

$$\text{angle} = \tan\text{-}1(yx) * (180/)$$

$$\text{velocity} = x2 + y2$$

So at last we need an algorithm which deals with ROS topics to accept goals using CLI and update the user through CLI. That code can go like -

```
defsendGoal():
    seq_ = int(input("Enter location number (0 to 6) (-1 to end process) - "))
    while(seq_ not in range(-1,7)):
            print("Invalid location number!!, Enter again.")
            seq_ = int(input("Enter location number (0 to 6) (-1 to end process) - "))
    if(seq_ == -1):
            empty_msg = GoalID()
            to_cancel_goal.publish()
            rospy.loginfo("----------- Bot Stopped -----------")
            rospy.loginfo("----------- Process Finished -----------")
            return
    poseStamp_ = PoseStamped()
    header_ = Header()
    pose_ = Pose()
    header_.seq = seq_
    header_.stamp = rospy.Time.now()
    header_.frame_id = "map"
    pose_.position.x = path[seq_][0][0]
    pose_.position.y = path[seq_][0][1]
    pose_.position.z = path[seq_][0][2]
    pose_.orientation.x = path[seq_][1][0]
    pose_.orientation.y = path[seq_][1][1]
    pose_.orientation.z = path[seq_][1][2]
    pose_.orientation.w = path[seq_][1][3]
    poseStamp_.header = header_
    poseStamp_.pose = pose_
    to_move_base.publish(poseStamp_)
    rospy.loginfo("----------- Path in progress -----------")
```

The above function asks the user which goal it wants to reach and publishes that goal to the map_server and move_base so that it gets executed.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 |

|| Volume 11, Issue 5, May 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1105148 |

IV. RESULTS

Six fixed places were used to test the robot in a unique world. In order to run the navigation stack on it, a costmap was first generated. Using teleop_twist_keyboard, the robot was controlled as the map was being created. The custom Python script was used to handle and publish goals. The location was accepted by that node from the user via the CLI (terminal). The robot must first receive one message on the automate/command subject before it will begin to take additional commands. The goals could be stopped by sending a stop message to the automate/command subject. They were published on the move_base_simple/goal topic. The same node on the move_base/result topic checked the success status.

The following table shows how the tests turned out. The experiment's positive outcomes and lower error rate were due to the use of Gazebo. However, the average inaccuracy was found to be 0.0732 m in the X direction and -0.0493 m in the Y direction.

	Given position (X,Y)	Achieved Position (X,Y)	Error in position
1	(-1.2809, 2.8191)	(-1.2885, 2.7746)	(0.0076, 0.0445)
2	(5.0868, 2.9511)	(5.0429, 2.9388)	(0.0439, 0.0123)
3	(5.4849, -2.9916)	(5.4479, -2.9694)	(0.0370, -0.0222)
4	(-2.6137, -4.7702)	(-2.5721, -4.7673)	(0.0416, -0.0029)
5	(-0.0370, -1.9030)	(-0.0791, -1.9237)	(0.0421, 0.0207)
6	(2.4604, -0.4160)	(2.4175, -0.4260)	(0.0429, 0.0100)
Average Error			(0.0358, 0.0187)

All values are in meter.

Conclusion and Future Work

The purpose of this software is to make robot autonomously navigate to its goal. While navigating it takes care of itself by avoiding the obstacles in its path. The path created while moving is the shortest one. The product can be used at any place where we need to send anything from one location to another location (depending on the thing which needs to be send) multiple times. E.g. warehouses. Or it can be used at universities or as a receptionist.

REFERENCES

1. [1] Rothong, Nuttapon&Thongchaisuratkrul, Chaiyapon. (2020). ROS Based Indoor AMR Design and Navigating Application. 291-294. 10.14416/c.fte.2020.03.042
2. [2] Y. Du, C. Ai and Z. Feng, "Research on navigation system of AMR based on ROS," 2020 IEEE International Conference on Real-time Computing and Robotics (RCAR), 2020, pp. 117-121, doi: 10.1109/RCAR49640.2020.9303274
3. [3] Koseoglu, Murat &Celik, Orkan&Pektas, Omer. (2017). Design of an autonomous mobile robot based on ROS. 1-5. 10.1109/IDAP.2017.8090199.
4. [4] Guo, P., Shi, H., Wang, S., Tang, L., & Wang, Z. (2022). An ROS Architecture for Autonomous Mobile Robots with UCAR Platforms in Smart Restaurants. Machines,10(10). https://doi.org/10.3390/machines10100844
5. [5] L. Cao and X. Zhu, "An Autonomous Service Mobile Robot for Indoor Environments," 2020 Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC), 2020, pp. 8-15, doi: 10.1109/IPEC49694.2020.9115180.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 |

|| Volume 11, Issue 5, May 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1105148 |

6. [6] P. Balamurugan, K. Harikrishnan, K. Kumaraguru, 2015, Mobi Robot Control using Internet of Thing (IOT), INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) NCEASE – 2015 (Volume 3 – Issue 22)
7. [7] Khassanov, Alisher& Alexander, Krupenkin&Alexandr, Borgul. (2015). Control of the Mobile Robots with ROS in Robotics Courses. Procedia Engineering. 100. 10.1016/j.proeng.2015.01.519

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor: 8.379

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details