Husky Robot Gazebo Simulation

[1]Y.Lokeswari, [2]P.Neeraja, [3]K.Meghana, [4]P.Mounika

[1,2,3,4] UG Student, Department of Electronics and Communication Engineering,
Dr K V Subba Reddy College Of Engineering For Women, Kurnool, Andhra Pradesh, India

Article Info

Page Number: 496-501

Publication Issue:

Vol. 71 No. 2 (2022)

Article History

Article Received: *25 December 2021*

Revised: *20 January 2022*

Accepted: *24 February 2022*

Publication: *28 March 2022*

Abstract

Husky is a robust unmanned ground vehicle intended for outdoor robotics research. It can conquer snow, sand, mud, and steep hills thanks to its powerful motors and all-terrain tires. The number and quality of robots, which make work easier and faster, have increased exponentially over the past decade. The study of robot Guidance, Navigation, and Control systems is the objective of this project. The work done to use ROS to simulate the Husky robot navigating to a series of waypoints in the Gazebo simulation environment is shown in this presentation. For the purposes of mapping and localization, SLAM will be utilized. ROS, which is currently one of the most widely used tools for programming robot software, will be used to carry out this simulation. Stereo cameras, GPS, LIDAR, and IMUS are a few of the features that can be added to the Husky robot.

1. Introduction

Measurements, sample collection, and access to difficult terrain have all been demonstrated by unmanned systems. Active research areas include the creation of new systems and their application in real-world settings. Early deployments of track-equipped remotely operated vehicles allowed for basic surveys and simple tasks to be carried out in risky industrial settings like nuclear power plants and risky natural settings like volcano calderas. Hybrid robots to study the Amazon rain forest and autonomous surface vessels to monitor water resources are examples of such systems.

Anomalies and faults can be found and where they are by using robotic systems. Two different robots, like unmanned ground vehicles (UGVs) and UAVs, can work together to get better results in terms of time lines and measurement accuracy. The limitations of aerial vehicles, which can travel quickly but have limited payload capacities, mission durations, and environmental interaction capabilities, have been addressed. Additionally, there are exclusion zones near airports, buildup areas, and restricted airspace for unmanned aerial vehicles (UAVs). On the other hand, although UGVs are less mobile, they are well-suited to transporting large payloads like specialized sensors and additional batteries or other power sources for extended endurances.

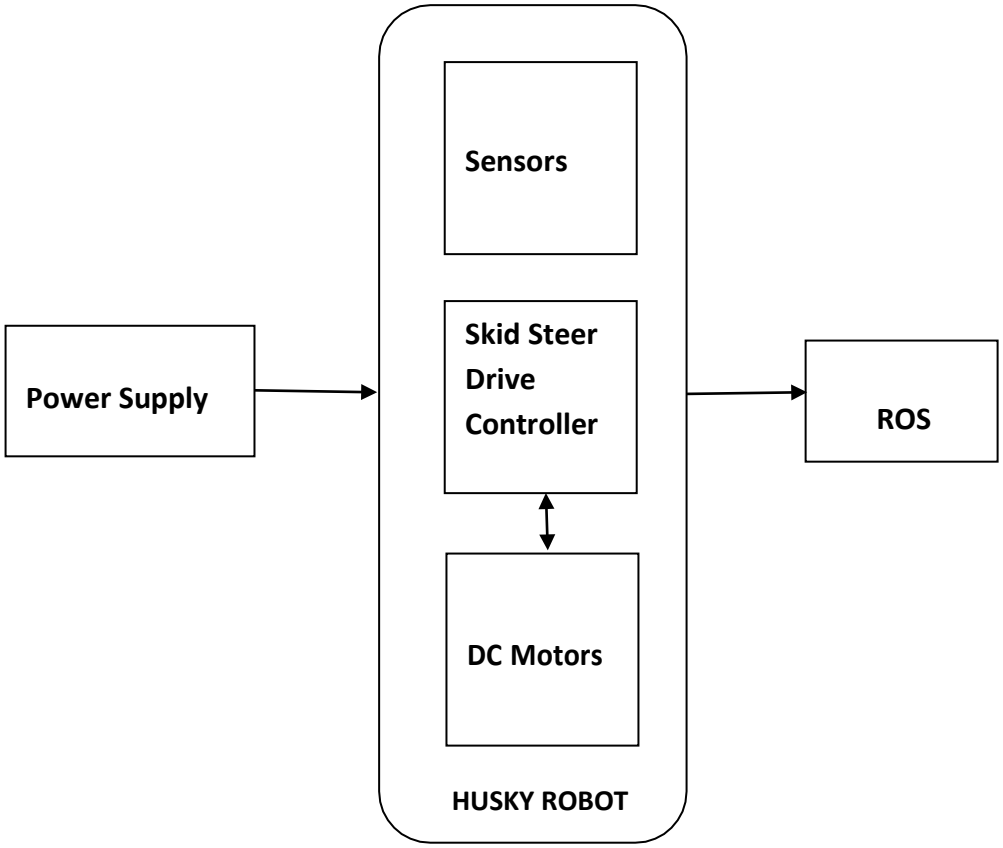

Fig.1 Block Diagram of Husky Robot Gazebo Simulator

2. Literature Review

Husky is a medium-sized platform for developing robotics. It is a unmanned ground vehicle (UGV) made for conducting robotics research in harsh outdoor settings. It can carry a wide range of payloads that can be tailored to meet specific research requirements thanks to its large payload capacity and power systems. Stereo cameras, GPS, LIDAR, and IMUs are a few of the features that can be added to the UGV. Husky was chosen for this project because it is effective at developing a variety of research types in a relatively straightforward and cost-effective manner. In addition, this robot's features can be modified quickly and easily. This adaptability in customization saves a lot of time and headaches when dealing with unexpected issues and changes. Husky's ease of use is yet another reason to use it. Husky makes use of an open-source serial protocol and is fully supported in ROS. Using existing research, such as Rugged and All-Terrain: This is extremely helpful for expediting the production of research results. Husky has few moving parts and is an elegantly simple design made of long-lasting materials. Husky is able to tackle challenging real-world terrain thanks to its high-performance, maintenance-free drive train and large lug-tread tires. It will allow for productive research for many years.

A Reliable Standard: Hundreds of researchers and engineers around the world rely on Husky. Husky has served as the test setup for numerous research papers. The established standard for new robot research and development efforts is Husky.

Control with precision: Husky's extremely high-resolution encoders provide enhanced capabilities for state estimation and dead reckoning. Even at slow speeds (1 cm/s), a finely tuned but user-adjustable controller provides extremely smooth motion profiles and excellent disturbance rejection.

Customizable:The robot experts can help you select and integrate payloads then configure the robot at the factory. Husky is plug-and-play compatible with our wide range of robot accessories and our system integrators will deliver a fully integrated turn-key robot

Fig.2 Husky Robot

3. Proposed System

A flexible framework for creating robot software is the Robot Operating System (ROS). The goal of this collection of tools, libraries, and conventions is to make it easier to create robust and complex robot behavior for a wide range of robotic platforms. because it is difficult to develop software for truly robust, all-purpose robots. From the robot's point of view, issues that humans consider to be insignificant frequently exhibit wildly divergent patterns across tasks and environments. It is impossible for any individual, laboratory, or institution to attempt to deal with these variations on their own because they are so difficult.

Consequently, ROS was designed from the ground up to foster software development collaboration in robotics. One laboratory, for instance, might have specialists in mapping indoor environments and contribute a world-class mapping system. Another group might be proficient at navigating with maps, and yet another group might have developed a computer vision method that is effective at identifying small objects in a mess. As is explained throughout this website, ROS was made specifically for groups like these to work together and build on each other's work.

The robotics middleware suite known as Robot Operating System (ROS) is free and open source. Despite the fact that ROS is not an operating system but rather a collection of software frameworks for the development of robot software, it offers services for a

heterogeneous computer cluster like hardware abstraction, low-level device control, the implementation of commonly used functionality, message passing between processes, and package management. A graph architecture depicts running sets of ROS-based processes. Processing takes place in nodes that can receive, post, and multiplex control, state, planning, actuator, and other messages. ROS is not a real-time operating system (RTOS) in and of itself, despite the significance of reactivity and low latency in robot control. However, real-time code can be integrated with ROS. ROS 2.0, a major revision of the ROS API that will take advantage of cutting-edge libraries and technologies for core ROS functionality and add support for real-time code and embedded hardware, addresses the lack of support for real-time systems.

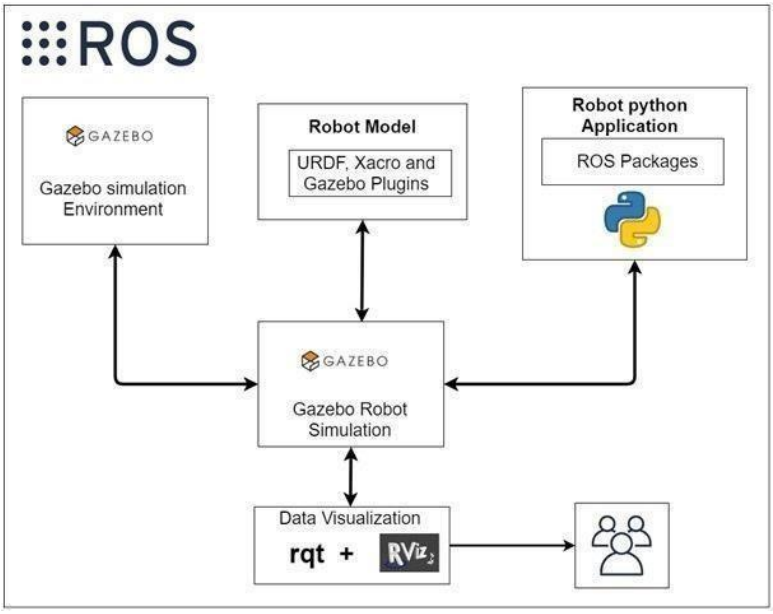

Fig.3 Gazebo Simulation

Due to their dependence on a large number of open-source software dependencies, the main ROS client libraries are designed for Unix-like systems. Ubuntu Linux is listed as "Supported" for these client libraries, while the community supports Fedora Linux, macOS, and Microsoft Windows as "experimental" versions. However, these restrictions are not present in the native Java ROS client library, rosjava, which has made ROS-based software for the Android operating system possible. Rosjava has also made it possible for ROS to be incorporated into a MATLAB toolbox that is officially supported and can be used on Linux, macOS, and Microsoft Windows. Additionally, roslibjs, a JavaScript client library that enables software integration into a ROS system via any standards-compliant web browser, has been developed.

ROS, or Robot Operating System, is a suite of open-source middleware for robotics. Despite the fact that ROS is not an operating system but rather a collection of software frameworks for the development of robot software, it offers services for a heterogeneous computer cluster like hardware abstraction, low-level device control, the implementation of commonly used functionality, message passing between processes, and package management. A graph architecture depicts running sets of ROS-based processes. Processing takes place in nodes

that can receive, post, and multiplex control, state, planning, actuator, and other messages. ROS is not a real-time operating system (RTOS) in and of itself, despite the significance of reactivity and low latency in robot control. However, real-time code can be integrated with ROS. ROS 2.0, a major revision of the ROS API that will take advantage of cutting-edge libraries and technologies for core ROS functionality and add support for real-time code and embedded hardware, addresses the lack of support for real-time systems..

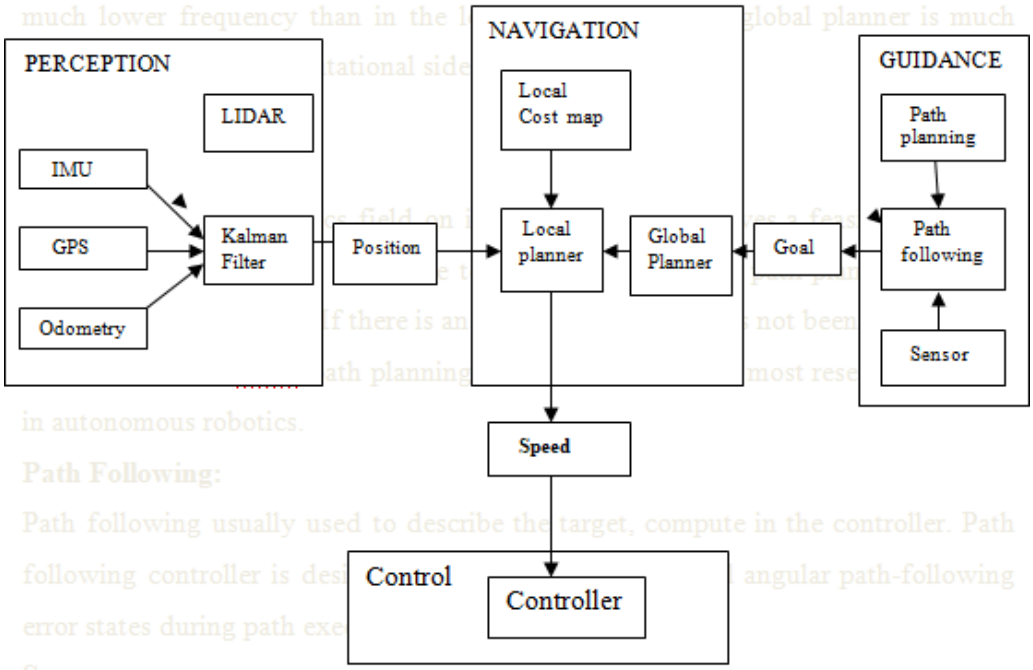

Fig.4 working of Gazebo

4. Conclusion

Today, robotics is a much more diverse field with numerous applications than it was even a decade or two ago. It can be challenging to keep up with the most recent advancements due to the ongoing improvements. Despite this, autonomous mobile robots appear to have a bright future in today's world due to the wide range of applications they can serve and the solid foundation of research and knowledge they are supported in developing and putting into use. ROS seems to have a bright future as well. People can begin using ROS in a short amount of time thanks to the open source of packages and information, which also provides new researchers with examples that can be very helpful.

The information and fundamental ideas for comprehending and simulating the mapping and localization (SLAM) navigation of a Husky robot in ROS have been presented in this report. In conclusion, this project has been very successful. Our simulated Husky is able to navigate through various waypoints, just like in the Simulation section, avoiding all kinds of obstacles and mapping the previously unknown environment along the way. This could be an important issue when operating a robot in the real world. Also, this report can help people get a basic

understanding of ROS and a few of its many applications, like guiding a Husky robot through an unknown environment and mapping it.

References

1. M. W. Spong and M. Fujita, "Control in Robotics," 2011.
2. Clearpath Robotics, "Husky. Unmanned Ground Vehicle.," [Online]. Available: https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/.
3. Robotics Business Review, "The 2017 RBR50 List Names Robotics Industry Leaders, Innovators," 2017. [4] Open Source Robotics Foundation, "About ROS," [Online]. Available: https://www.ros.org/about-ros/. [5] A. Martinez and E. Fernandez, Learning ROS for Robotics Programming, 2013.
4. J. M. O'Kane, A Gentle Introduction to ROS, Jason Matthew O'Kane, 2013.
5. R. Siegwart, I. Nourbakhsh and D. Scaramuzza, Introduction to Autonomous Mobile Robots, Segunda ed., The MIT Press, 2011.
6. S. Riisgaard and M. Blas, "SLAM for Dummies: A Tutorial Approach to Simultaneous Localization and Mapping," 2005. [Online]. Available: http://ocw.mit.edu/courses/aeronauticsand-astronautics/16-412j-cognitive-robotics-spring-2005/projects/1aslam_blas_repo.pdf.
7. J. Fabro, R. Longhi, A.Scheneider and T. Becker, "ROS Navigation: Concepts and Tutorial," in The ROS Multimaster Extension for Simplified Deployment of Multi-Robot Systems, Springer International Publishing, 2016, pp. 121-160.
8. ROS, "ROS.org," [Online]. Available: http://wiki.ros.org/move_base.
9. ROS, "gmapping," ros.org, [Online]. Available: http://wiki.ros.org/gmapping.
10. L. Joseph and J. Cacace, "Building a map using SLAM," in Mastering ROS for Robotics Programming, Second ed., 2018.
11. [13]T. Kamegawa, N. Sato, M. Hatayama, Y. Uo, and F. Matsuno, system integration for grouped rescue robots system using robohoc network. Workshop on robots and sensors integration in future rescue Information system ROSIN'10(2010).