ROS-BASED ROVER WITH 3D MAPPING AND USER INTERFACE

Joel Georgie Jacob
MGP20URB038
Saintgits College of Engineering

Lakshya sharma
MGP20URB044
Saintgits College of Engineering

Neo Thomas Padiyara
MGP20URB052
Saintgits College of Engineering

John Sajeeve
MGP20URB039
Saintgits College of Engineering

Er. Pratap Pillai
Assistant Professor
Saintgits College of Engineering

***Abstract*—This project aims to create a versatile and robust robotic platform that can autonomously navigate through unknown environments while providing real-time 3D mapping feedback. The Robot Operating System (ROS) serves as the foundation for the project, allowing seamless integration of modules, sensors, and algorithms. The rover is equipped with state-of-the-art sensor technologies, such as a Kinetic sensor, which captures the surrounding environment's depth and color information. The data is processed using point cloud processing algorithms, allowing for visual simultaneous localization and mapping (SLAM) techniques to construct a real-time 3D representation of the rover's surroundings.**

The success of this project will offer practical applications in fields like exploration, search and rescue operations, and surveillance. The integration of ROS, 3D mapping, and user interface design in advanced robotic systems contributes to the advancement of autonomous navigation, mapping technologies, and human-robot interaction, opening up new avenues for robotic applications in real-world scenarios.

***Index Terms*—ROS, Four-wheeled rover, 3D Mapping, RTAB-Map, Rocker Bogie Mechanism, User-Interface, Kinect Sensor**

I. Introduction

The Robot Operating System (ROS) has revolutionized robotics and autonomous systems development. The ROS-based rover, equipped with 3D mapping and a user interface, offers precise navigation, comprehensive mapping, and intuitive interaction. This innovative technology allows developers to seamlessly integrate hardware, sensors, and algorithms, resulting in a highly adaptable and customizable system. The ROS-based rover generates detailed 3D maps in real-time, aiding navigation and providing valuable information about terrain, obstacles, and objects. The user interface, typically graphical, allows operators to visualize the rover's surroundings, monitor its status, and control its movements. The integration of ROS in rover design offers numerous advantages, including the utilization of existing ROS packages, libraries, and tools, accelerating development, simplifying the integration of additional functionalities, and fostering collaboration and knowledge sharing among researchers and developers.

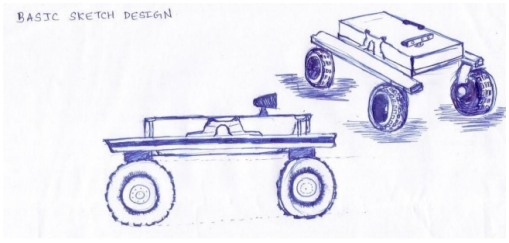

Fig. 1. Initial sketch design

II. Literature Survey

A. On the Mobility of All-Terrain Rovers.

A kinematic model for articulated mobile robots and a quasi-static model of forces acting on the rover. These techniques are used to evaluate the locomotion performance of all-terrain rovers employing rocker-type suspension systems, The kinematic model is used to formulate a mathematical description of the rover's motion, while the quasi-static model is used to analyze the forces acting on the rover as it moves over uneven terrain.

B. Study on the use of Microsoft Kinect for robotics applications.

Strong sunlight is a huge issue since it interferes with the Kinect infrared sensor by blinding it with too much infrared radiation. On the other hand, Kinect sensors are effective in detecting items at night that are undetectable during the day. Extremely efficient for robotic applications requiring indoor navigation. The fact that the Kinect sensor is so inexpensive is a huge benefit. The inexpensive Kinect sensor makes it possible to purchase an additional sensor that uses stereo

vision to enhance 3D identification in point clouds, particularly outside.

C. Using the Kinect as a navigation sensor for mobile robotics.

3D SLAM is a process of feature matching that uses odometry to achieve position estimates. It is implemented using RGB-D SLAM which performs visual SLAM using feature-matching algorithms. This process is very slow and resource intensive, but offers advantages such as 3D model building, pure visual SLAM, a lower price, and the inclusion of color into maps. Kinect is a viable option for use as a sensor for mobile robotic navigation and SLAM.

D. Indoor mapping using Kinect and ROS.

It contains the utilization of ROS, TurtleBot 2, and Kinect operation, The packages used for this study are Gmapping, Interactive Marker, and Exploration. It is also found that the Kinect sensor produces a more accurate map compared to non-filtered laser range finder data.

E. 3-D mapping with an RGB-D camera.

The visual SLAM scheme is applied to the real-time creation of indoor three-dimensional maps using consumer-level depth sensors, and the accuracy of the built indoor maps is evaluated by coordinate transformation. This scheme can solve the problem of real-time construction of indoor maps in the surveying and mapping field because the camera collects many feature points in the course of movement, and the efficiency of building three-dimensional point cloud maps is not high. Laser and visual sensors can be used to explore the indoor environment in the later stage, so as to obtain rich and comprehensive environmental information and establish a semantic map of an indoor environment.

F. Design of a transformable mobile robot for enhancing mobility.

The six-wheel structure platform has high mobility and stability, and the rocker arm system is regulated by feedback information from the sensor, which is beneficial for traversing road surfaces and maintaining balance. After completing the vehicle structure and installing the sensing equipment, the robot must pass a field test to prove the functional feasibility of the vehicle. The robot was mainly assessed in the surrounding environment of the school campus. The environment includes one-way roads, sidewalks, slopes, and indoor areas. Through these different field tests, the stability, control actions, and perceptions are verified in order to demonstrate the application of this robot.

III. Challenges We Faced While Making

A. Mechanical Related Problems

- When it came to purchasing the necessary components, we were unable to find them conveniently in Kerala, so we acquired them from several websites.
- We were unable to find a box that met our specifications, so we settled on a PCB Enclosure.

B. Design Related Problems

- Design flaws in the motor mount that came with the purchased motors caused the motors to become loose from their position while operating the rover.
- Problems with the Rocker Bogie mechanism arise from limited designs for four wheels, as most designs were related to six wheels.

C. Programming Related Problems.

- Some of the drivers were outdated or lacked proper support for implementing the camera and the Kinect sensor with Raspberry Pi.
- We also faced problems while we were transferring the workload of 3D Mapping from the Raspberry Pi to the laptop.
- We also faced several difficulties importing the camera feed while creating the user interface for end-user interaction using React.

IV. System Model

The system model will encompass the integration of hardware components, software modules, and communication protocols required for the functioning of the 3D mapping rover.

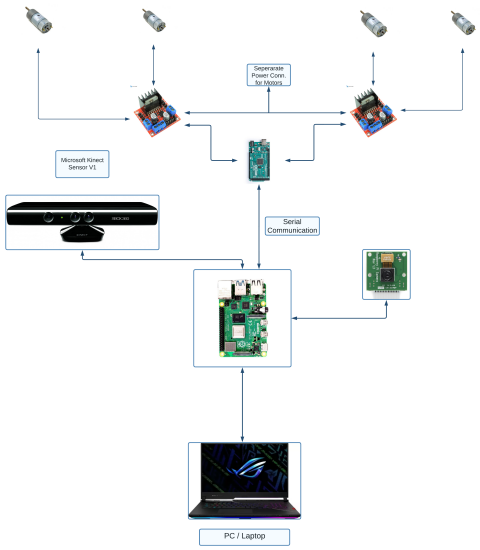

Fig. 2. System Model

A. Architecture of ROS (Robot Operating System)

ROS Noetic is an evolution of ROS, incorporating improvements and new features while maintaining compatibility with existing codebases. Its modular and distributed architecture enables efficient development and integration of robotic systems. Nodes are executable programs that perform specific tasks and communicate with each other through publishing and subscribing to messages on named topics. Topics facilitate publish-subscribe communication, while messages define data structures exchanged between nodes. Services provide

a synchronous request-response communication mechanism between nodes, allowing for advanced behaviors. Packages are self-contained units of code and resources, while the ROS Master is a central component for node discovery and communication. Tools like roscore, roslaunch, Rviz, and rqt are used to promote modularity, flexibility, and ease of development.

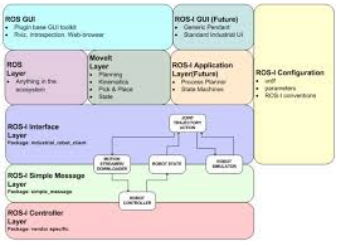

Fig. 3. Architecture of ROS

B. Architecture and Pin-out Diagram Arduino Mega

The Arduino MEGA is a microcontroller board based on the ATmega2560 chip, designed for projects requiring more computational power or larger inputs and outputs. It features a high-performance 8-bit AVR microcontroller with 256KB of flash memory, 8KB of SRAM, and 4KB of EEPROM. The board has 54 digital and 16 analog inputs, with 15 of them being PWM outputs. It also includes multiple communication interfaces, a flexible power supply system, and support for programming and development through the Arduino Integrated Development Environment (IDE). The MEGA also includes headers and connectors for expansion and customization, making it an ideal choice for projects requiring more I/O connections, enhanced computational capabilities, and expanded memory resources.

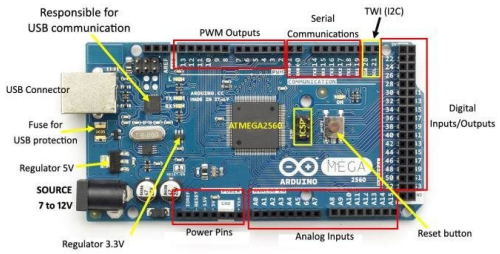

Fig. 4. Architecture and Pin-out Diagram Arduino Mega

C. Architecture and Pin-out Diagram Raspberry Pi 4

The Raspberry Pi 4 is a versatile and powerful single-board computer with a powerful architecture consisting of key components: CPU, GPU, RAM, Storage, and Connectivity. It features a Broadcom BCM2711 SoC and quad-core ARM Cortex-A72 processor, with a VideoCore VI GPU with OpenGL ES 3.0 support. RAM options include 2GB, 4GB, and 8GB, while storage is provided by a microSD card slot. Connectivity options include USB 2.0 and USB 3.0 ports, dual-band Wi-Fi, Bluetooth 5.0, dual monitor output, GPIO (General-Purpose Input/Output) with a 40-pin GPIO header, and a 5V power supply. This architecture enables users to develop a wide range of projects and applications, ensuring a high-performance computing experience.

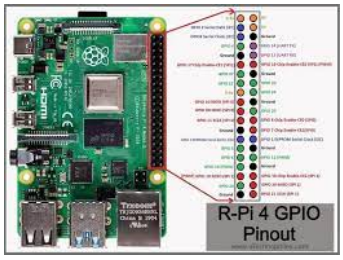

Fig. 5. Architecture and Pin-out Diagram Raspberry Pi 4

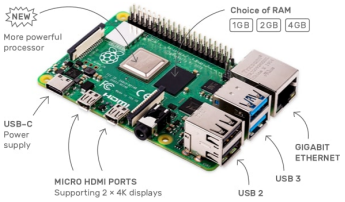

Fig. 6. Raspberry-PI-Board

D. Architecture of Microsoft Kinect Sensor V1

The Microsoft Kinect Sensor V1, also known as the Kinect for Xbox 360, is a cutting-edge depth-sensing technology that revolutionized motion and gesture-based interaction. Its architecture combines an RGB camera, an infrared projector, and an IR sensor in a compact device, capturing standard color images and generating a depth map of the scene. This information, combined with RGB data, enables real-time tracking of users' movements and gestures, making it useful in various applications such as gaming, virtual reality, and medical research. The Kinect's depth-sensing capabilities enable robots to perform tasks like object recognition, obstacle avoidance, and simultaneous localization and mapping (SLAM). The real-time depth data is particularly useful for creating detailed maps of the robot's surroundings, which are essential for navigation and autonomous decision-making. The user-friendly SDK simplifies integration with robot control systems, making it an accessible choice for enhancing the perceptual capabilities of robotic platforms.

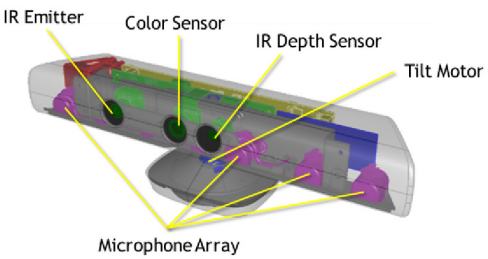

Fig. 7. Microsoft Kinect Sensor V1

E. CAD Models

Below are CAD models for our four-wheeled rocker bogie mechanism rover, along with mechanical parts models.:-

1) Differential Bar: The differential bar is a crucial mechanical component in a four-wheeled rocker-bogie rover, connecting the wheels on each side of the suspension system. It is an integral part of the rover's drivetrain mechanism and plays a vital role in smooth and efficient mobility. The rocker-bogie suspension system, developed by NASA's Jet Propulsion Laboratory, enables the rover to traverse uneven and rocky terrains while maintaining stability. The differential bar distributes power and torque evenly between the wheels, enabling independent wheel movement and effective power distribution. It also helps maintain traction on different terrain types by allowing wheels to rotate at different speeds, accommodating variations in ground contact and obstacles.

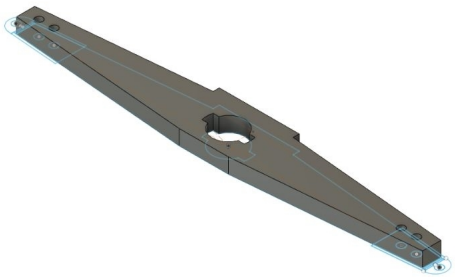

Fig. 8. Differential Bar

2) Pivot: The pivot of a rocker-bogie rover is the point where the rover rotates or turns. It adjusts the speed and direction of its wheels by varying the relative speed of the wheels on either side. The pivot typically lies between the middle pair of wheels. This feature enables four-wheeled rovers to navigate challenging terrain and execute precise maneuvers during scientific exploration missions.

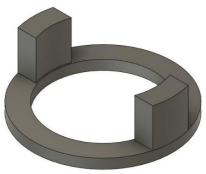

Fig. 9. Pivot

3) Tie Rod Joint: A tie rod joint in a four-wheeled rocker bogie rover is a mechanical connection that helps control the movement and steering of the rover's wheels. The rover's front and rear wheels, both larger and placed in tandem, are directly linked to the steering mechanism through these tie-rod joints. When these joints are actuated, they exert forces on the wheel hubs, allowing for precise control over the direction of these front and rear wheels. This design remains crucial for maintaining stability and ensuring accurate steering control, especially when navigating challenging terrains. They need to be sturdy, adjustable, and capable of withstanding the forces and stresses encountered during rover operations.

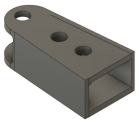

Fig. 10. Tie Rod Joint

V. System Software Requirements

To get adequate outcomes throughout project work, the system requires some software. Some of the software that we required to run our four-wheeled rocker bogie rover are written below: -

A. Arduino IDE

The Arduino Integrated Development Environment (IDE) is a crucial platform for programming Arduino microcontrollers, playing a pivotal role in our project. We used the IDE to write and upload code to the Arduino microcontroller, which is an integral component of our rover's motor control system. We also utilized the rosserial library, which allows seamless communication between the Arduino and the Robot Operating System (ROS) environment. This integration established a robust link between the rover's microcontroller and the central ROS system, ensuring precise and real-time motor control. The Arduino IDE and rosserial library provided a user-friendly interface for code development and seamless integration of hardware and software components in our project.

B. Linux Ubuntu 20.04.6 LTS

The operating system was Ubuntu 20.04.6 LTS, a popular Linux release. The selection of Ubuntu was deliberate, owing to its great support for ROS, which has made it the go-to operating system for robotics applications. We were able to smoothly integrate numerous ROS packages and libraries into this operating system, allowing our rover to maneuver, map its surroundings in three dimensions, and communicate with people via a user-friendly interface.

C. ROS (Robot Operating System)

ROS, or Robot Operating System, is an open-source framework that revolutionizes robotics and automation. It offers a comprehensive collection of tools and libraries for complex systems, making it a popular choice among academics and engineers worldwide. ROS simplifies sensor integration, hardware control, communication, and task organization. Its modular architecture and community support make it a viable platform for various robotic applications. ROS served as the foundation for our project, integrating sensors like Kinect Sensor V1 and Raspberry Camera V1 for seamless data transmission and control. It also enabled real-time 3D mapping

for autonomous navigation and a user-friendly interface for remote monitoring and control, showcasing ROS's adaptability in improving robotic applications.

D. RTAB-Map

RTAB-Map (Real-Time Appearance-Based Mapping) is an open-source Simultaneous Localization and Mapping (SLAM) framework that is extensively used in robotics and autonomous systems. It was important in revamping our rover's mapping capabilities in our project. We were able to construct a very comprehensive and accurate 3D map of the rover's surroundings in real time by integrating RTAB-Map. This allowed us to traverse and explore strange areas while also building a thorough 3D depiction of our environment. As we moved the rover, RTAB-Map fused data from the Kinect Sensor V1 to build and enhance the 3D map. This map served as a foundation for future research and decision-making inside the rover's control system, as well as navigational assistance.

E. Libfreenect Driver

The libfreenect driver, a userspace driver for the Microsoft Kinect V1 from OpenKinect, was critical to our project. This open-source driver connects the Kinect Sensor V1 to our Raspberry Pi 5, allowing for easy connection and data sharing. We enabled our rover to gather depth and RGB information from the Kinect sensor by using the features of the libfreenect driver, allowing for real-time 3D mapping of the rover's environment.

F. React

React, a popular JavaScript package for generating user interfaces offers a flexible and fast foundation for developing interactive online apps. React was critical in designing the user interaction experience in our project. We used React to create a user-friendly and straightforward interface that allows for continuous contact with the rover. We were able to create a responsive and dynamic interface using React, allowing users to manage and monitor the rover's operations. The incorporation of React into our project considerably improved the usability of the rover, guaranteeing that users can interact with it efficiently for navigation, data analysis, or mission planning.

VI. Implementation

A. Hardware Design and Assembly

A model was sketched based on the specified dimensions of the rover. After going through several designs from the web, we later finalized the model design after designing it in Fusion 360. We later assembled the physical rover as per the design specifications

B. ROS Packages

After uploading the motor control code to Arduino Mega via Arduino IDE. We created a ROS Workspace and added the necessary packages for implementing the Kinect Sensor V1, and Raspberry Pi Camera V1. Later on, we created a launch file so that we could execute all the packages for rover control with a single command in the terminal.

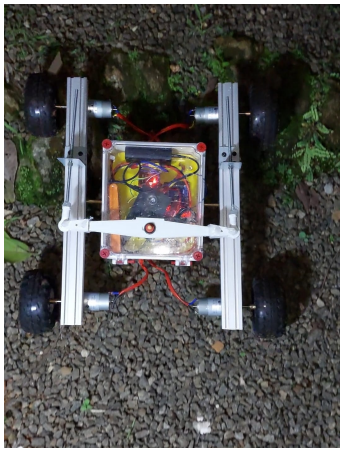

Fig. 11. Field Testing of the Rocker-Bogie Design

C. 3D Mapping

The process of 3D Mapping was implemented using RTAB-Map (Real-Time Appearance-Based Mapping), and we imported its necessary ROS Packages from GitHub. We launch the RTAB-Map in a seperate terminal whenever we want to initialize the mapping process, as it launches a GUI with the live mapping feed as we control the rover

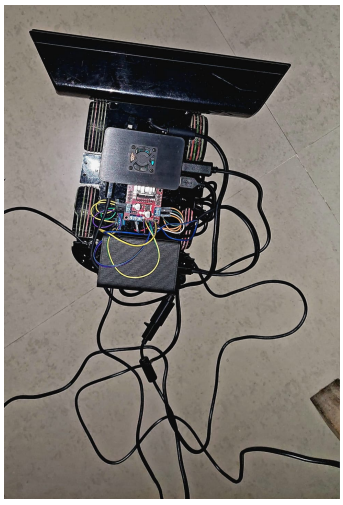

Fig. 12. Prototype for testing sensor and motor control

D. User Interface

The user interface was designed using React. In the interface, we have implemented a status bar showing whether we are connected to the rover or not and a joystick to control the rover.

Fig. 13. ROS User-Interface using React

VII. Experimental Analysis, Results And Future Scope

A. Result And Future Scope

In conclusion, our mini project focused on developing a ROS-based rover equipped with 3D mapping capabilities and a user interface. The project successfully achieved the objective of creating a functional 3D map of the rover's surroundings. However, the limitations of the Raspberry Pi's computation power were evident as the mapping process was time-consuming. Additionally, the need for redesigning the motor mount to enhance ground clearance became apparent during the project.

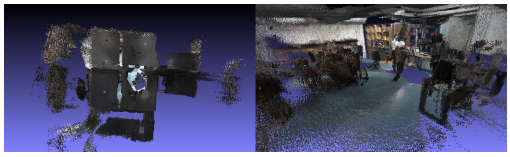

Fig. 14. 3D Map of FabLab (1) Top View (2) Inner View

Despite these challenges, the project provided valuable insights into the integration of hardware, software, and robotics principles. The successful implementation of 3D mapping and the identification of areas for improvement underscored the practical complexities involved in creating sophisticated robotic systems.

The future scopes of this project include the following:

- Performance Optimization: Optimize the 3D mapping efficiency by addressing computation limitations, exploring hardware upgrades, and reducing time.
- Mechanical Enhancement: Redesign motor mount for increased ground clearance, improving rover mobility and mapping accuracy.
- Autonomous Navigation: Integrating autonomous navigation capabilities by incorporating sensors like LIDAR, cameras, and IMU, so the rover could navigate and map its environment independently, reducing the need for constant user intervention.
- Robotic Manipulator Integration: Expanding the project to include robotic manipulator arm challenges kinematics, control, object manipulation, and collaborative tasks.
- Machine Learning Integration: Implementing machine learning techniques for better decision-making, obstacle avoidance, and path planning could enhance the rover's overall performance.
- User Interface Refinement: Improving the user interface for remote control and visualization of mapping data could enhance user experience and make the system more user-friendly.

B. Code

https://github.com/jj7258/ROS-Based-Rover-With-3D-Mapping-and-User-Interface

VIII. Acknowledgment

The satisfaction that accompanies the successful completion of any task would be incomplete without acknowledging the people whose constant guidance and encouragement has crowned all the efforts with success. It would not have been possible to complete the project without their valuable help, cooperation, and guidance. First, I thank God Almighty, for lending the talent to do this project and completing the work successfully.

I take this opportunity to thank Dr. Josephkunju Paul. C, Principal, Saintgits College of Engineering, Pathamuttom for providing the best facilities and environment to prepare and present this project. I am grateful for his support and cooperation. I also thank Dr. Sreekala K. S., Associate Professor and Head of the Department, of Electronics Engineering, for her advice and encouragement. I express my sincere gratitude to Dr. Sita Radhakrishnan (Mini Project Coordinator) and Er. Pratap Pillai (Project Guide), for their helpful guidance, suggestions, and encouragement in the completion of the project. Finally, I extend my special thanks to our other staff members and to all my beloved friends for their timely help.

References

[1] G. Reina and M. Foglia, "On the mobility of All-Terrain Rovers," Industrial Robot: An International Journal, vol. 40, no. 2, pp. 121–131, Mar. 2013, doi: https://doi.org/10.1108/01439911311297720.

[2] R. A. El-laithy, J. Huang and M. Yeh, "Study on the use of Microsoft Kinect for robotics applications," Proceedings of the 2012 IEEE/ION Position, Location and Navigation Symposium, Myrtle Beach, SC, USA, 2012, pp. 1280-1288, doi: 10.1109/PLANS.2012.6236985.

[3] Oliver, Ayrton and Kang, Steven and Wünsche, Burkhard and Macdonald, Bruce. (2012). Using the Kinect as a navigation sensor for mobile robotics. ACM International Conference Proceeding Series. 509-514. 10.1145/2425836.2425932.

[4] H. I. M. A. Omara and K. S. M. Sahari, "Indoor mapping using Kinect and ROS," 2015 International Symposium on Agents, Multi-Agent Systems and Robotics (ISAMSR), Putrajaya, Malaysia, 2015, pp. 110-116, doi: 10.1109/ISAMSR.2015.7379780.

[5] Endres, F., Hess, J., and Sturm, J. (2017), 3-D mapping with, an RGB-D camera, IEEE Transactions on Robotics, Vol. 30, No. 1, pp. 177

[6] Kim, W. Jeon, and H. Yang, "Design of a transformable mobile, robot for enhancing mobility," International Journal of Advanced, Robotic Systems, vol. 14, no. 1, p. 1729881416687135, 2017

[7] Anil Mahtani, L. Sanchez, E. Fernandez, A. Martinez, and L. Joseph, ROS programming : building powerful robots : design, build, and simulate complex robots using the Robot Operating System. Birmingham: Packt Publishing, 2018.

[8] M. Quigley, B. Gerkey, and W. D. Smart, Programming robots with ROS. Sebastopol: O'Reilly and Associates Incorporated, 2015

[9] . L. Joseph, Learning robotics using Python : design, simulate, program, and prototype an autonomous mobile robot using ROS, OpenCV, PCL, and Python. Birmingham, Uk: Packt Publishing, 20