Programming and Control of Physical Autonomous Robots via ROS 2*

Lili Ma[1], Christian Rosa[1], Xiaohai Li[1], Yu Wang[1]
Benito Mendoza[1], Andy S. Zhang[2]
[1]Deptartment of Computer Engineering Technology
[2]Deptartment of Mechanical Engineering Technology
CUNY-New York City College of Technology
New York, NY 11201

Abstract

This paper describes two exemplary projects on physical ROS-compatible robots (i.e., Turtlebot3 Burger and Waffle PI) for an undergraduate robotics course, aiming to foster students' problem-solving skills through project-based learning. The context of the study is a senior-level technical elective course in the Department of Computer Engineering Technology at a primarily undergraduate teaching institution. Earlier courses in the CET curriculum have prepared students with programming skills in several commonly used languages, including Python, C/C++, Java, and MATLAB. Students' proficiency in programming and hands-on skills make it possible to implement advanced robotic control algorithms in this robotics course, which has a 3-hour companion lab session each week.

The Robot Operating System (ROS) is an open-source framework that helps developers build and reuse code between robotic applications. Though mainly used as a research platform, instructors in higher education take action in bringing ROS and its recent release of ROS 2 into their classrooms. Our earlier work controlled a simulated robot via ROS in a virtual environment on the MATLAB-ROS-Gazebo platform. This paper describes its counterparts by utilizing physical ROS-compatible autonomous ground robots on the MATLAB-ROS2-Turtlebot3 platform.

The two exemplary projects presented in this paper cover sensing, perception, and control which are essential to any robotic application. Sensing is via the robot's onboard 2D laser sensor. Perception involves pattern classification and recognition. Control is shown via path planning. We believe the physical MATLAB-ROS2-Turtlebot3 platform will help to enhance robotics education by exposing students to realistic situations. It will also provide opportunities for educators and students to explore AI-facilitated solutions when tackling everyday problems.

1 Introduction

Robotics engineering is a multidisciplinary field built upon electrical, mechanical, and computer engineering. It deals with designing, building, operating, and engineering robots and robotic systems based on theoretical understanding and practical application. From its inception, robotics has been an inherently interdisciplinary field, bringing together diverse domains such as engineering, cognitive science, computer science, and knowledge from social sciences and humanities [9]. When teaching robotics in higher education, it is thus important to keep up with the latest developments in robotics as well as many related fields such as Artificial Intelligence (AI), data science, computer vision, Internet of Things (IoT), and Cybersecurity.

The context of this study is an undergraduate robotics course offered as a technical elective in the Department of Computer Engineering Technology (CET) at a primarily four-year teaching institution. Preceding courses in the CET curriculum have equipped students with knowledge and skills in mechatronics, embedded systems, programming, and cyber-physical systems. This robotics course aims to provide hands-on experience working with complex computer-controlled systems that integrate physical components such as sensors and actuators.

Autonomous mobile robots with a simple arm on top were used as the physical robotic platform for this course, built from VEX robotic kits using the Cortex microcontroller that comes with the kit [5, 6]. Driven by the need to also serve the recently approved Software Engineering Technology (SET) curriculum, we have been exploring robotic systems that allow for complex software and hardware integration.

This paper describes the development of two exemplary projects using the Turtlebot3 robots. The experimental setup involves a host computer running MATLAB and a physical robot (Turtlebot3 Burger or Waffle PI), each being a ROS node on the ROS network. ROS stands for the Robot Operating System [11], which is a set of open-source software libraries and tools that help researchers and robotic engineers build robot applications. ROS has been widely used by researchers and developers to build and reuse code between robotics ap-

plications. However, due to the demanding requirements of C++/Python/Java programming skills and familiarity with Linux, the adoption of ROS in an undergraduate curriculum is still rare. Recently, MathWorks released its ROS Toolbox [2], making it easier to interact with both simulated robots [3] and physical ROS-supported robots (see Fig. 1).

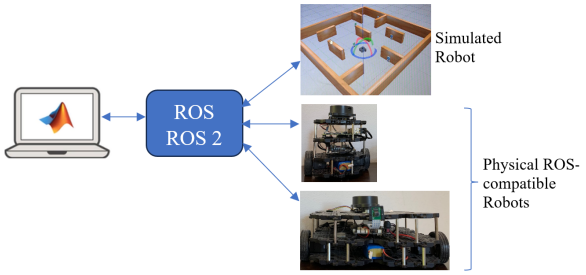


Figure 1: Connecting to ROS robots using MathWork's ROS Toolbox.

Our earlier work in [8, 7] explored the usage of ROS (particularly, ROS 1) for programming and control of a simulated robot in the Gazebo simulation environment [Fig. 2 (left)]. This paper presents its counterpart and extension, which programs and controls a physical ROS-compatible robot (Turtlebot3 Burger or Waffle PI) on ROS 2 [Fig. 2 (right)]. Note that ROS 2 is the second generation of ROS representing a step forward in the robotic framework [12]. While getting access to the robot's onboard sensors may be different due to the available topics and their types, control algorithms that were developed for the simulated robot can be readily applied to the physical robots, demonstrating the re-usability of the codes & algorithms.

The physical robotic platform, i.e., the MATLAB-ROS2-Turtlebot3 platform, utilizes benefits from both sides of MATLAB and ROS 2:

- The physical ROS-compatible robot provides students with experience in authentic Linux operating systems and ROS 2 programming.
- MATLAB already has many other toolboxes dedicated to education and research (such as computer vision toolbox, artificial intelligence toolbox, and machine learning toolbox). These toolboxes will significantly shorten the algorithm development curve.
- Both Mathworks and ROS have well-established mechanisms for developers to share their codes and experience. This allows students to use support and resources from the community.

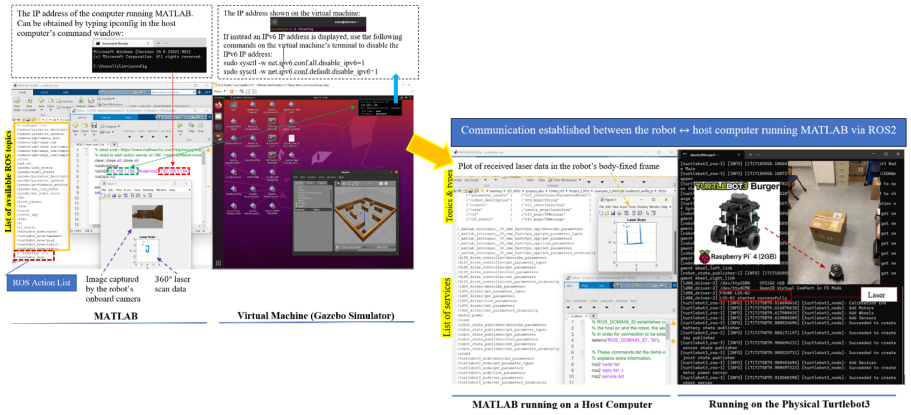


Figure 2: Using MATLAB to control the simulated robot (left) and physical robot (right).

- Students can collect and save data from the robot's onboard sensors during classes. They can conduct post-processing and refine their algorithms outside of the classroom.

The objective of this paper is to explore the feasibility of using the MATLAB-ROS2-Turtlebot3 platform to enhance robotic education and undergraduate research. The subsequent sections of this paper describe the operation on the robot side, communication with and control of the robot issued from the host computer (the MATLAB side), and two exemplary projects that incorporate sensing, perception, and control.

2 Turtlebot3 Startup and Operation

Before implementing our algorithms to control the robot, we initially learned how to operate it by running the pre-installed programs. It is worth noting that the results outlined in this paper were achieved on Turtlebot3 Burger, but all algorithms can be applied to the Turtlebot3 Waffle PI due to ROS compatibility. As an illustration, the following presents the results of running the pre-installed SLAM program on Burger and provides the necessary sequence of operations under ROS 1.

SLAM, short for Simultaneous Localization And Mapping, is a common topic for autonomous mobile robots, as seen in commercially available robots like robot vacuums. The Turtlebot3 Burger employs SLAM using its onboard 2D range sensor, the LiDAR (Light Detection and Ranging), mounted on top of

the robot. The LiDAR rotates 360 degrees at a speed of 6 rotations per second. It emits a laser beam to measure the closest objects at roughly an incremental angle of 1.5 degrees. The data gathered in this process can be used to discern between walls and other objects, aiding Turtlebot3 in map construction and navigation.

Figure 3 shows the results of the robot performing SLAM in a room, where the green points denote LiDAR scan data, the black area represents walls, and the grey area shows the movable area (i.e., open space).

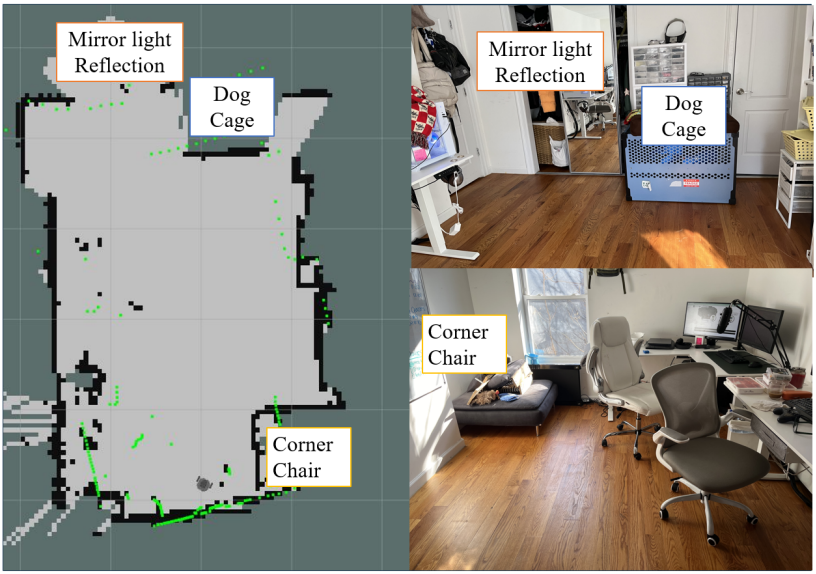


Figure 3: Turtlebot3 SLAM.

The Turtlebot3 operates on Linux. Following the installation instructions as given in Turtlebot3's e-manual [10], a server version is installed on the Raspberry PI on the robot. To use any graphic visualization tools, we installed a Virtual Machine (VM) on the host computer to run the desktop version of Linux. A Virtual Machine is a digital replica of a physical computer, enabling one to emulate Linux on a Windows system. We then need to select the ROS version. As each ROS version runs on different Linux versions and many ROS versions were no longer maintained, we opted for ROS 1 Noetic, which allows fully functioning SLAM and Navigation features as shown in Fig. 4.

To initiate the SLAM task, we began by establishing the communication/-connection between Turtlebot3 and the host computer by configuring them on

the same network. We then executed the startup sequence on the Turtlebot3 robot. On the host computer, we started the Virtual Machine, from where we run three terminals each of which handles different tasks:

- Referring to Fig. 4(a), the terminal located in the upper left connected to the Turtlebot3 robot. This terminal facilitated a wireless connection to the Turtlebot3 Operating System, enabling us to operate the robot without requiring peripherals like a monitor, mouse, and keyboard.
- The terminal positioned in the upper right served as the 'roscore' terminal. 'roscore' encompasses a set of nodes and essential programs necessary for a ROS1-based system. Under ROS 1, it is necessary to have 'roscore' up-running to facilitate communication among ROS 1 nodes.
- The terminal located at the bottom is dedicated to Teleoperation, a pre-installed program. This terminal enabled us to navigate the robot using the keyboard on the Virtual Machine.
- The above three terminals collectively enable the operation of the Turtlebot3. An additional terminal will initiate the SLAM functionality [Fig. 4(b)].

(a) Startup (b) SLAM

Figure 4: Turtlebot3 startup and operation on ROS 1.

As shown in Fig. 4, using pre-installed programs, we have successfully controlled the Turtlebot3 Burger to move and navigate in an environment, collect laser data, and use the collected laser data to aid in mapping and navigation. These investigations demonstrate that the robots were correctly assembled and that all operating systems, libraries, and software were properly installed. We are now ready to control the robots using our programs.

3 Control the TurtleBot3 using MATLAB

The MATLAB-ROS2-Turtlebot3 experimental platform was set up by following the Quick Start Guide of the ROBOTIS e-Manual [10] for the construction and installation of the robot. On the host computer that runs MATLAB,

MathWork's ROS Toolbox is required and installed. Successful installation and communication between these two ROS nodes is shown in the right figure in Fig. 2. Some details are:

- On the robot's processor (Raspberry PI), we installed Ubuntu Server 22.04 and ROS 2 Humble Hawksbill. We used SSH to access the Raspberry PI and brought up basic packages to start TurtleBot3 applications.
- The ROS domain ID on MATLAB needs to be set the same as that of the robot so that these two can establish a connection in between. For example, we used 30.
- Once communication is established (by using the same Domain ID), the command "ros2 node list" lists all nodes on the ROS 2 network; "ros2 topic list -t" lists all available topics and their types; and "ros2 service list" lists all available services. These commands help to confirm that communication/connection has been successfully established.
- Figure 5 shows sample MATLAB codes that subscribe to topics (odometry and laser scan data), as well as establish a publisher that modifies the robot's linear & angular velocities.
- Visualization of the ROS 2 graph is displayed via the "rqt" tool.

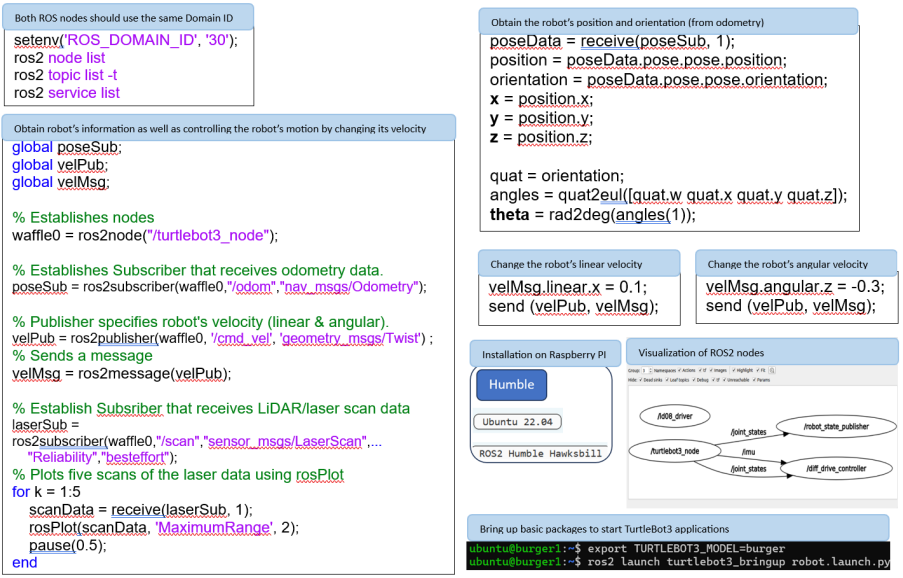


Figure 5: Sample MATLAB codes.

After successfully obtaining raw sensor data from the robot and modifying its behavior, the next task is to develop algorithms for perception and decision-making. This will be done on the MATLAB side, as demonstrated via two exemplary projects in Secs. 4 and 5.

4 Path Planning

This project provides a simple scenario for students to implement algorithms that allow an autonomous mobile robot to work in an unknown environment. The robot is assumed to have an onboard range sensor. Particularly for the Turtlebot3 robots, its 360 Laser Distance Sensor (LDS-02) is a 2D laser scanner capable of sensing 360 degrees around the robot. Initially, the robot will explore its surroundings, getting to "know" its environment by sensing and recording its laser data.

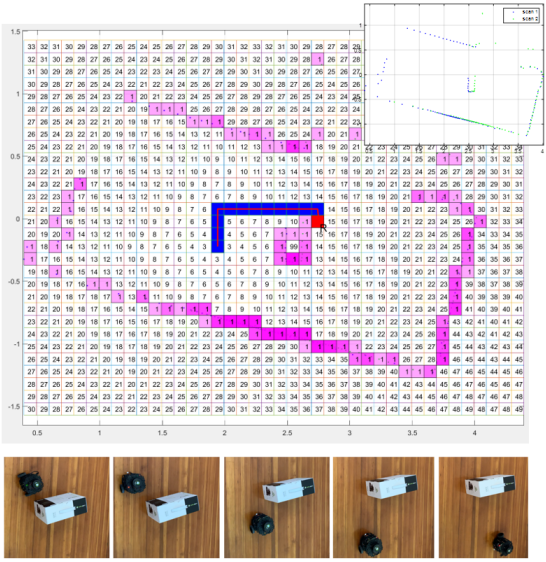


Figure 6: Path planning using Wavefront algorithm.

Areas where objects are detected will be considered as "occupied" or "inaccessible." After getting familiar with its surroundings, the robot is expected to find its way to a specified goal location without collision with obstacles. Solutions to this path-planning problem provide the robot with an obstacle-free path.

The Wavefront algorithm is the most basic but powerful approach to tackling path planning. The workspace is modeled as a 2D grid map. Locations occupied by the detected objects are marked inaccessible by denoting their values as "1". Grids representing open spaces will be assigned a non-zero value, following the algorithm's policy. Eventually, after all the open spaces have gotten their values updated/assigned, the robot will find its way from its current location (denoted by the red grid in Fig. 6) to the specified goal location by counting down the value one less than before. Several snapshots of the robot's movement as it follows the path are also shown.

Through this simple path planning project, the MATLAB-ROS-Turtlebot3 platform demonstrates its capability in fast algorithm development and implementation for processing large amounts of data.

5 LiDAR Data Processing

Transformation of laser data to the inertial frame: In many cases, data needs to be integrated. For example, in the path planning project described in Sec. 4, the robot collects laser data in several locations to obtain a more comprehensive idea about its environment. Laser data is collected in the robot's body-fixed frame. To transform these data to the same frame, i.e., the inertial frame, the robot's position and orientation at the time of sensing need to be used. This is a good example for students to understand the importance of homogeneous transformation, a common topic in robotics. Figure 7 shows an illustration of integrating three individual laser scans to form a better representation of the robot's environment.

The path planning algorithm described in Sec. 4 used raw laser data without going through additional processing. This project aims to familiarize students with advanced perception processes, focusing on object and shape recognition using laser scan data, specifically rectangle fitting. Based on data preprocessing (to obtain the point cloud to be processed), in the next, we will describe our simple segmentation (also called cluster detection) and rectangle-fitting schemes.

Distance-based segmentation: To conduct segmentation of 2D point clouds, the distance between consecutive points is typically used to determine if they are part of the same object. Using distance as the criterion, laser points are grouped as one cluster if they are close to each other (i.e., the distance in between is less than a pre-specified threshold). For each laser point, we first compute the minimal distance between this point with all existing clusters. If the minimum of these distances is greater than the specified threshold, this point is considered to belong to a new group. Otherwise, it is assigned to the one closest to it.

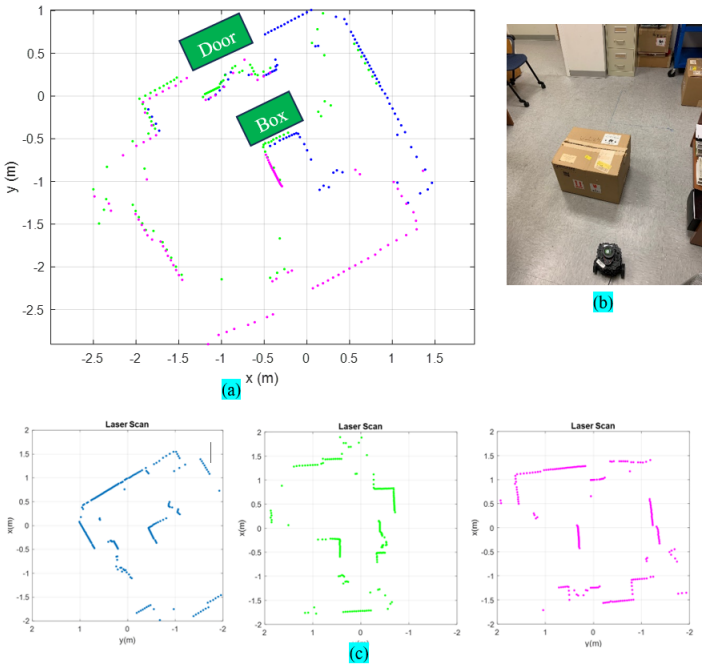


Figure 7: Transformation of laser data to the inertial frame, a good example of applying homogeneous transformation.

Figure 8 highlights the effect of selecting different parameters/values. Data points that were thought to belong to the same object are plotted in the same color. Different clusters are plotted in different colors. An index is written next to each cluster. It can be seen that setting the thresholds higher will result in fewer clusters.

Rectangle fitting to L-shape data: After data segmentation and clustering, we performed rectangle fitting to each cluster. We computed the sum of distances (i.e., errors) from each point in this cluster to the fitted model. If this sum of errors is "small" enough and the width & length of the fitted model are "large" enough, we will consider this cluster of points to represent a rectangle. The motivation for considering this L-shape rectangle fitting problem is to derive vehicle pose estimation in the Advanced Driving Assistance Systems (ADAS) scenario [13] in the future.

Figure 9 gives an illustration of the rectangle-fitting process. The raw laser data is shown in (a). Data segmentation is given in (b), which outputs 6 identified clusters. Rectangle fitting via MATLAB's LiDAR point cloud

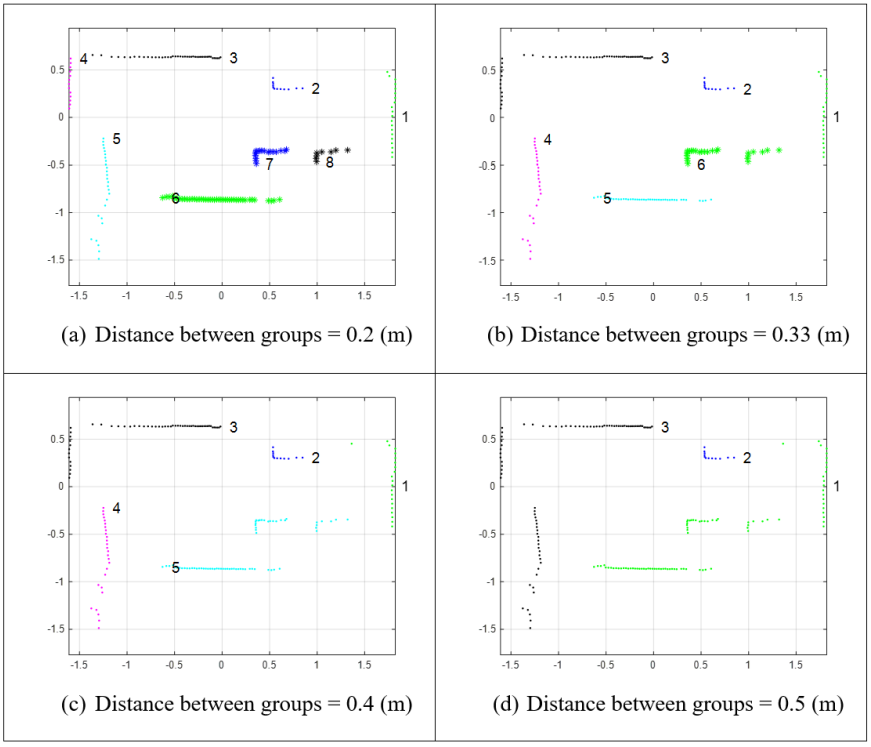


Figure 8: Clustering algorithms applied to the obstacle points using distance-based segmentation. The minimal distance between groups is set to be 0.2, 0.33, 0.4, and 0.5 in (a), (b), (c), and (d), respectively.

analysis tools (particularly the pcfitcuboid() function) is used to fit a rectangle to each cluster [4]. The sum of the minimal distance from each point in the cluster to the fitted model is computed using the files shared on MATLAB's File Exchange Center, i.e., the "Distance from points to polyline or polygon" routines [14]. The final fitted rectangles are displayed in (b). Calculation of the distance error is shown in Fig. 9(c), that is, to compute the distance between each point (denoted by a pink cross) to its closest point (denoted by a red dot) on the rectangle.

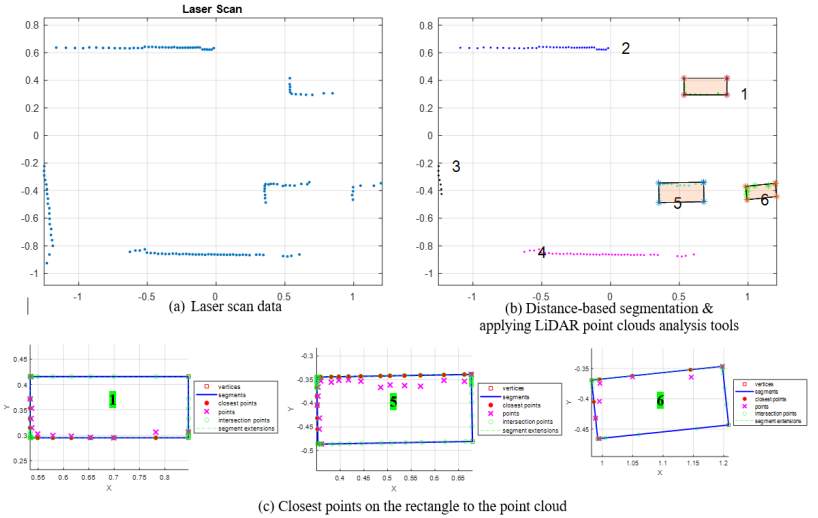
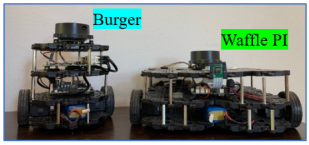


Figure 9: Rectangle fitting routine by applying distance-based segmentation and MATLAB LiDAR point cloud analysis tools.

6 Discussion, Conclusion, and Future Work

Adopting proper simulation and/or experimental platforms in a robotics course is essential for students to apply classroom knowledge to practical situations. Due to the widespread use of ROS in research and industry, as well as the emerging integration of Artificial Intelligence (AI) with robotics and the need to align with a new Software Engineering Technology (SET) curriculum, we are currently investigating different options to update the existing physical robotic systems used in an undergraduate robotics course (CET 4952: Robotics Technology). The Turtlebot3 robots are excellent candidates for various features:
Being low-cost, small-size, sturdy, and portable. A low price will allow the department to purchase enough number of sets. Currently, we are thinking of providing one robot to each group consisting of two students. Being small-sized will allow the robots to be stored in cabinets inside the laboratory for students to check out and then check in during the lab sessions. Being sturdy will allow these robots to be used semester after semester since the robotics course is offered in both the spring and fall semesters. Being portable will allow these robots not to interfere with other courses that use the same lab.
Being able to provide students with hands-on experience in two fundamental areas (Autonomous Mobile Robots and Robotic Manipu-

lator): For undergraduate robotic education, it is a good practice to introduce these two areas either in one course or a sequence of two courses. So, we are looking for robotic systems that can cover both. The Burger and Waffle PI robots will allow students to fully explore almost all aspects essential to autonomous mobile robots, including sensing, perception, laser data processing, image processing, computer vision, map building, navigation, path planning, coordinated control, and integration of AI. The 5-DOF robotic manipulator, as shown in Fig. 10, will help students reinforce their understanding of homogeneous transformation, forward and inverse kinematics, and trajectory generation. Placing the robotic arm on top of the Waffle PI robot results in a robotic system that combines mobility with action (Fig. 10). The results presented in this paper focused on autonomous mobile robots. Future investigations will be conducted for the 5-DOF robotic arm and the integrated robotic system.



Autonomous Mobile Robots


5-DOF Robotic Manipulator


A mobile base carrying a robotic arm

Figure 10: Autonomous mobile robots and robotic manipulators. The second and third pictures are from ROBOTICS website [10].

Being open-source and having well-established mechanisms for sharing and support within the community/society: Robotics is a rapidly evolving field that intertwines with many other areas such as electronics, communication, cybersecurity, computer science, signal & image processing, and mathematics. Advancements in these closely related areas will in turn have huge impacts on robotics. Algorithms and methods that enhance the autonomy of robots are developed much faster than traditional sources such as textbooks and conference proceedings. Researchers, educators, and students begin to use "new" ways to obtain timely support and keep up with the most recent developments in this field. Open source, which has become a trend since the last decade, has transformed into a global tendency, especially in fields like robotics. The demand for open source makes ROS-compatible robots more intriguing since ROS is a set of open-source software libraries and tools. The ROS forum allows users to ask questions, comment on others' discussions, and thus provide/receive support from the community. Similarly, MathWorks' File Exchange Center allows one to share/post their developments. The MATLAB-

ROS2-Turtlebot3 experimental platform utilizes benefits from both ends, including algorithm development, community support, and ready adoption of the developed algorithms to other robots.

Being able to serve as an undergraduate research platform and integrate the AI computational system with a robot: Due to the rapidly evolving nature of the robotics field, research needs to be seamlessly integrated with teaching to expose students to the latest developments. Exponential increases in computing power, sensor actuators, and communication transceivers have made producing robotic systems economically feasible. Students can now get access to fully-functioning robots at a much cheaper cost. Research opportunities thus become more available to undergraduates. We think robotics-related activities (courses and projects) should provide a propelling force in promoting undergraduate research. Further, in response to the emerging trend of using AI to find better solutions, integrating AI with robotics should play a leading role since this integration is inherent and embedded [1]. For example, existing feature extraction and face recognition algorithms in computer vision already have AI flavors. The distance-based segmentation and rectangle-fitting routines as described in Sec. 5 could be improved by AI-facilitated adaption and self-learning in determining the specified thresholds (i.e., the minimal distance among groups and the lower/upper bounds of the rectangles' dimensions).

The two exemplary projects presented in this paper confirmed the usage of the MATLAB-ROS2-Turtlebot3 robotic platform in a robotic course, by providing engaging and leaning environment through realistic scenarios. They will also help others to develop and teach similar robotic courses, which is inline with the nation's trend in Artificial Intelligence & Machine Learning curriculum. The physical platform can help boost undergraduate research by allowing students to explore AI-facilitated solutions to improve the robot's functionalities including mapping, navigation, obstacle avoidance, and coordination. In future investigations, we will explore: a) vision-based control via the onboard Raspberry PI camera; b) control of the 5-DOF (degree-of-freedom) robotic arm for pick-and-place tasks and then the integrated robotic system for warehouse applications; and c) formation control of multiple Robots.

7 Acknowledgement

The National Science Foundation, Award 2240516, supported this research.

References

[1] Robert Avanzato. "Deep Learning Projects for Multidisciplinary Engineering Design Students". In: *ASEE Annual Conference and Exposition*. 2023.

[2] Robert Avanzato and Culllen Wilcox. “Introductory Mobile Robotics and Computer Vision Laboratories Using ROS and MATLAB”. In: *ASEE Annual Conference and Exposition*. 2018.

[3] Siavash Farzan. “Project-Based Learning for Robot Control Theory: A Robot Operating System (ROS)-Based Approach”. In: *ASEE Annual Conference and Exposition*. 2023.

[4] Felipe24 Jiménez and Miguel Clavijo. “LiDAR point clouds analysis computer tools for teaching autonomous vehicles perception algorithms”. In: *Computer Applications in Computer Education* (Feb. 2024), pp. 1–17.

[5] Lili Ma. “Teaching Undergraduate Robotic Courses using Enhanced VEX Robots”. In: *Journal of STEM Education: Innovations and Research* (July 2021).

[6] Lili Ma et al. “Development of a Raspberry PI-Controlled VEX Robot for a Robotics Technology Course”. In: *ASEE Annual Conference and Exposition*. June 2023.

[7] Lili Ma et al. “Introducing ROS-Projects to Undergraduate Robotic Curriculum”. In: *ASEE Annual Conference and Exposition*. June 2023.

[8] Lili Ma et al. “Online Robotics Technology Course Design by Balancing Workload and Affect”. In: *Transactions of the SDPS: Journal of Integrated Design and Process Science* (Jan. 2022).

[9] O. Michalec, C. O’Donovan, and M. Sobhani. “What is robotics made of? The interdisciplinary politics of robotics research”. In: *Humanities and Social Sciences Communications* 65.8 (Mar. 2021), pp. 1–15.

[10] Robotics. *Turtlebot3 e-Manual*. `https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/`.

[11] ROS. *Robot Operating System*. `https://ros.org/`.

[12] S. Macenski et al. “Impact of ROS 2 Node Composition in Robotic Systems”. In: *IEEE Robotics and Autonomous Letters* (2023).

[13] Xiaotong Shen, Scott Pendleton, and Marcelo H. Ang Jr. “Efficient L-shape Fitting of Laser Scanner Data for Vehicle Pose Estimation”. In: *IEEE Conference on Robotics, Automation and Mechatronics*. July 2015.

[14] Michael Yoshpe. *Distance from points to polyline or polygon*. `https://www.mathworks.com/matlabcentral/fileexchange/12744-distance-from-points-to-polyline-or-polygon`.