Teaching Autonomous Navigation Using an Open-Source Middleware in a Hybrid Format

Abhishek Patil and Jungyun Bae
Michigan Technological University

Abstract

Along with other topics in mechatronics, autonomous navigation has been advanced amazingly in recent decades and is now playing a pivotal role in many industrial automation applications. This paper discusses the new strategies designed to teach autonomous navigation of mobile robots using robot operating system (ROS), the most popular open-source robotics middleware. At Michigan Tech, the authors offered the course to undergraduate seniors and graduate students in a hybrid format with live broadcasted recitations and in-person labs. As the time and capacity of the lab were limited, the instructors put their effort into providing resources that students could work on implementations on their own. The main difference from the past teaching was providing Gazebo simulation environments and template codes with ROS. Based on the recitations taught before the lab sessions, the students are asked to write the core of the algorithms in Python using the provided templates. Providing simulation tools helped the students implement and debug their algorithms before testing their codes on the actual robots. While students could practice and experience autonomous navigation on their own through the simulations, they could experience the difference between the simulator and the real-world robot motions. Many industries already use ROS for their implementation in the field, and many commercial robots are available for ROS. Therefore, the authors believe this newly designed course provides a more profound knowledge and practical experience in mechatronics.

Introduction

Autonomous navigation for mobile robots is one of the leading technologies that accelerated the fast advancement of mechatronics in recent decades. Autonomous navigation plays an essential role in many industrial automation applications such as warehouses and manufacturing lines. While the number of students interested in robotics and mechatronics increases, few universities offer courses that can have hands-on experience with both simulation and real-world robots. Fortunately, Michigan Tech offers students as many courses as possible that include hands-on experience on the curriculum for both undergraduate and graduate students. It is evident that more hands-on experiences on techniques that follow the current industry trends can help the students better understand their job opportunities. For this reason, we designed a new course that focuses on hands-on lab sessions utilizing the most widely used open-source middleware, ROS.

ROS [1] is a middleware that provides libraries and tools to help software developers create robot applications. Many state-of-art algorithms are developed in ROS and open to the public, which allow people to develop other necessary parts of the applications based on those algorithms. Many robots, sensors, and motors are already implemented in ROS, which is easier for the users to utilize the actual hardware and simulation. This aspect is beneficial not only in academics but also in industries. More industrial robots are also utilizing ROS for their testing and also in actual applications. As possible applications of robots in industries are increasing, the number of hiring robotics engineer and robotic software developers are also increasing. Having students' intense hands-on experiences on ROS and autonomous navigation techniques prepares them for the robotics industry. In addition to ROS itself, training the students in programming with Python also supports students in having coding skills to be ready for their work.

This paper shares a course designed to train students in autonomous navigation with ROS in a hybrid format. The course focuses on two-hour hands-on lab sessions while providing two times of one-hour recitation sessions each week. The course is designed to cover the introductory level of core knowledge for autonomous navigation and is targeted to bring more students interested in the area. The rest of the paper is as follows: we first introduce the course details; next, we discuss the successes and challenges; and finally, we conclude by summarize and discuss future directions.

The Course for Autonomous Navigation

Class Format

The three-credit course was offered for senior undergraduate students and graduate students. Due to the pandemic, the recitation lectures were performed in a hybrid format, both in-person and live streaming, while hands-on lab sessions were conducted in-person only. However, as the lab capacity was more limited due to social distancing, we provided simulation environments and coding templates before the lab sessions to allow students to practice and be ready before they came. Students participated in the lab as a team of two, so if one were under quarantine, the other could do the lab in person. If all team members were under quarantine, the teaching assistant ran the codes for the team on a real robot with live streaming by appointment. There were five Turtlebot3 [2] and Linux-based laptops for actual robot experiments. The environment for the experiments was provided weekly in the lab space based on the materials. Two exams on recitation sessions, the lab reports, and the final project (competition) took 30% of the grade in each. The last 10% was given based on participation, such as attendance and discussion.

Learning Objective

The course targets the students to program software that interacts with robotic hardware and implement algorithms that address fundamental problems to operate robotic and autonomous systems, including path planning, localization and mapping, perception, kinematics, and sensor fusion. It also aims for the students to work effectively in a team to solve challenging problems without clearly prescribed solutions.

Topics

The course covered the fundamental theories of autonomous navigation of ground mobile robots through recitation sessions. The topic includes programming basics in Python and ROS, sensors and actuators, locomotion, mobile robot kinematics, perception, vision, path planning, localization, and mapping. The lab sessions consist of nine themes, including some of the two-week assignments. The themes of the lab sessions are as follow:

Lab #	Topic
Lab 1	Play with Turtlebot3
Lab 2	Explore Gazebo with Turtlebot3
Lab 3	Dead reckoning
Lab 4	Robot parameter tuning
Lab 5	Wall following with LiDAR (2 weeks)
Lab 6	Perception with LiDAR
Lab 7	Perception with Vision
Lab 8	Path planning (2 weeks)
Lab 9	Localization and Mapping with LiDAR

Details of Recitation Sessions

The recitation sessions dealt with the fundamental theories in mobile robotics. Many of the students registered for the course were unfamiliar with Python. Thus, the recitation started by introducing Python programming basics and installing and launching ROS on their PCs before getting into the theories. From the second week, students learned about sensors commonly used for mobile robots, locomotion focused on different types of wheeled mobile robots, and mobile robot kinematics. In mobile robot kinematics, we concentrate on differential kinematics, degree of mobility and steerability, and configuration space. The midterm exam covered these materials to review what they had learned about the mechanical side of the robot itself. After that, we focused on core algorithms required for autonomous navigation, from extracting the necessary information from the sensors to path planning, localization, and mapping. For each topic, at least three different algorithms were introduced to the students while having one implemented during the lab sessions. The final exam covered autonomous navigation algorithms and core knowledge. While having 19 actively involved students throughout the semester, an average of five students attended the recitation sessions in person while other students attended through the live stream. As the semester proceeded, students were more involved in the recitation sessions, as the instructor asked both in-person and virtual students to answer more questions with example problems. Students asked questions both before and after the class, either regarding the theories or labs.

Details of Lab Sessions

Pre- and post-lab assignments were given to the students for each lab. Students were expected to open the provided simulation environment, plan their strategy for the given task of the lab, and try to implement it in the given simulation environment before they come to the lab as their pre-

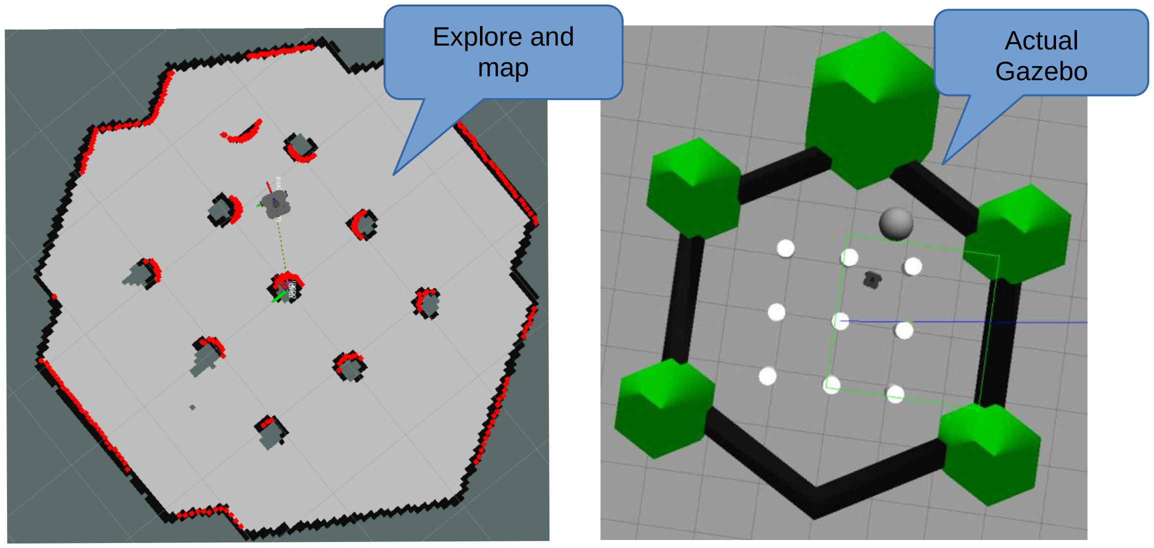

Fig. 1. The simulation environment for Labs 1 and 2. Students are asked to explore the environment by remotely control the given Turtlebot3 and observe the sensor data.

lab homework. Then, during the lab session, students applied their implementation results to the real robot and observed the performance. The post-lab assignment should include experimental results, analysis, and discussion. If students want to work on the lab further, two hours of the additional in-person lab were available by appointment.

The first two weeks focused on introducing ROS to the students expecting them to be familiar with the ROS environment and programming. While installing ROS on students' PCs went parallel, students were asked to launch the Gazebo simulation environment, the most common simulation tool correlated with ROS, and run open-source codes to explore ROS and Gazebo simulation as presented in Fig. 1. The advantage of using Turtlebot3 for this course was that the vendor was providing training materials. Thus, although the focus of the training was different, we could utilize some of the provided simulation environments and codes to introduce the software. Unfortunately, we had to explore only the simulation for the first two weeks due to restrictions on in-person classes.

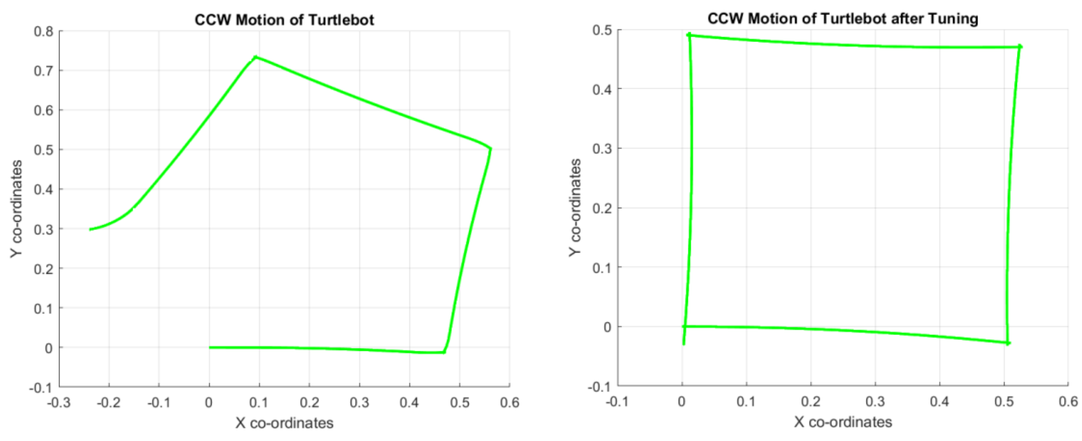

Fig. 1. One of the comparison results that students submit for Lab 4. Students calibrated the kinematic parameters through the observation of the performance that has distortions and errors. The robot departs from (0,0) and heads back to the origin through the square path.

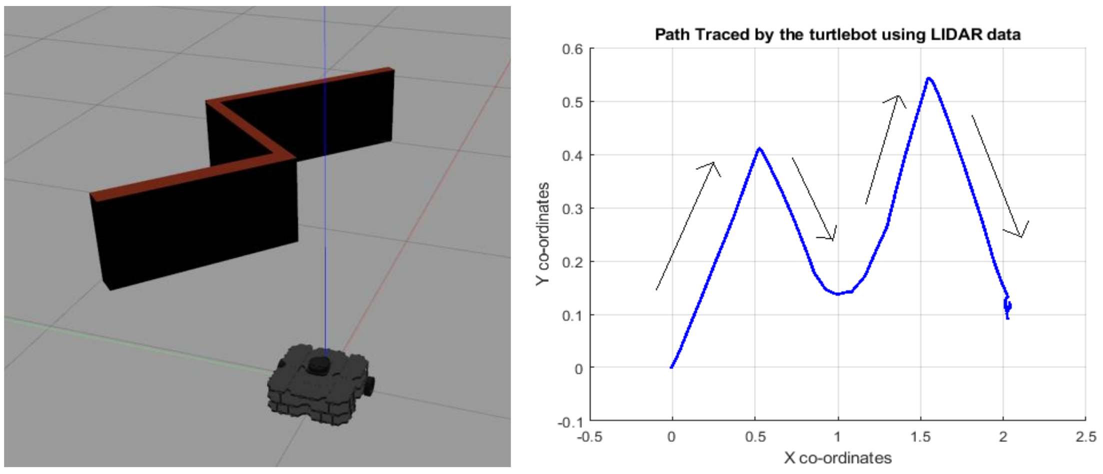

Fig. 2. The simulation environment for Lab 5 (left) and one of the wall-followed trajectories submitted by the students (right). Most students implemented a simple logic to maintain the minimum distance on the left side of the robot.

The third and fourth labs are closely connected. In Lab 3, students are asked to observe and record the robot's performance on tracking the square-shaped path with time-based control. Based on their calculation of linear and angular velocities required to follow the given route, students practice how to publish the velocity commands to the robot. Then, we asked the students to think about how to improve their performance. In Lab 4, based on the collected data in Lab 3, students calibrate some parameters that could affect the performance caused by internal (mechanical) distortions or errors. By comparing the before and after the calibration performance, students could observe the improvements, as shown in Figure 2.

In Lab 5, students implemented their algorithms to let the robot follow the nearby wall. The wall has two 90-degree corners, as shown in Figure 3, which were the challenging points for the students. As students did not learn the perception at this point, we have introduced how to utilize the range data to trace the wall with some manually coded commands.

Again, Labs 6 and 7 are closely connected as both labs deal with perception but with different sensors. In Lab 6, students implemented the algorithms they learned during the recitations to

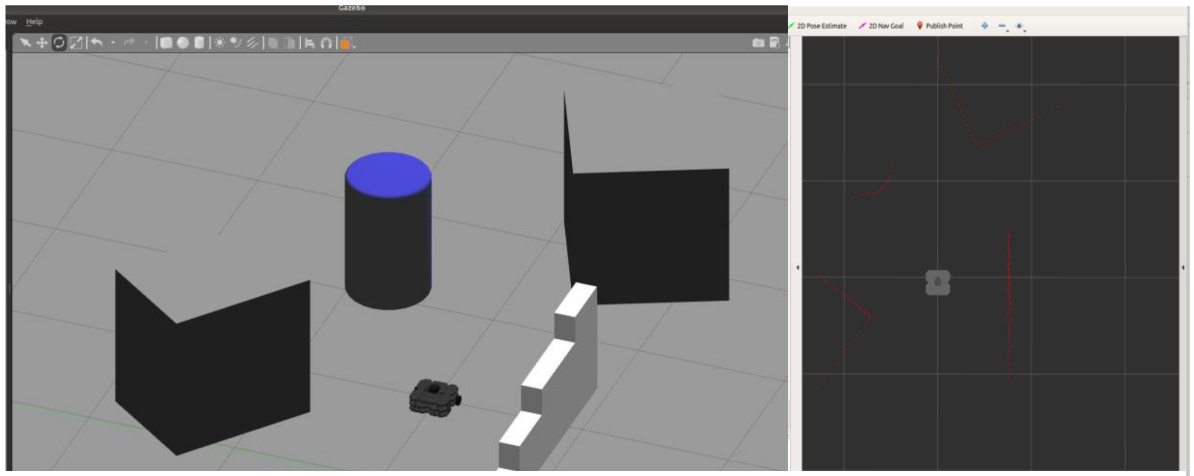

Fig. 3. The simulation environment for the perception with LiDAR. Students are asked to extract the features from the collected data and detect circular shape of the obstacle.

extract lines and circles from the range data. Most of the students applied Hough transform method for extractions. As the assignment, students are asked to find a cylinder and follow the wall of the cylinder. In Lab 7, using the OpenCV package, students practiced extracting color information from the image data transferred from the camera. In the end, students generated commands for the robot to detect the red cylinder and follow the cylindrical wall.

Lab 8 focused on path planning. While introducing various path planning algorithms, we focused on the potential field-based path planning for implementation. As students were overwhelmed with coding, the instructor presented the pseudo-code for the algorithm during the recitation. Students generated the path planner to generate the trajectory avoiding collision with any obstacles, given the initial and goal positions.

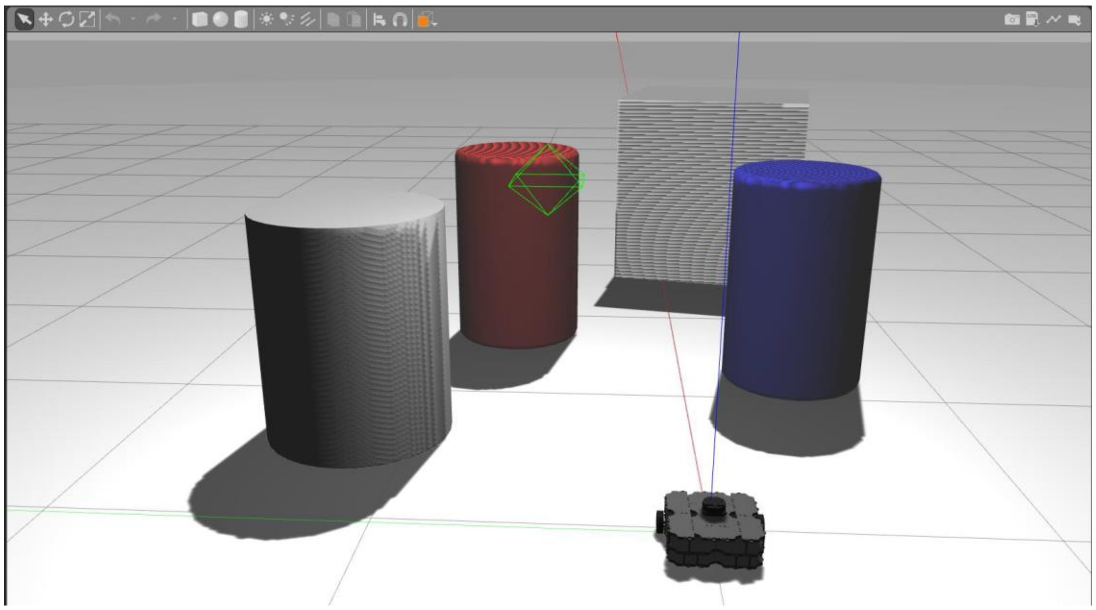

Fig. 4. The simulation environment for the perception with camera.

Lastly, in Lab 9, students explored the given maze utilizing open-source localization and map building packages. As students' average levels of coding were not high enough to implement localization and mapping algorithms on their own, various algorithms were introduced during the recitations while providing the open-source packages to the students to explore still how the algorithms work within the given environment.

Once students complete all nine labs, they are asked to integrate and improve their algorithms to participate in the final project competition. The competition consists of three stages. At the first stage, the robot needs to get out of the given maze-like environment. The robot is expected to plan a path and follow it within the fastest time without collision with the walls. At the second stage, the robot is expected to plan a path based on the map and LiDAR sensor measurements and reach the goal position without collision with the moving obstacle, another Turtlebot. Although we have not directly implemented this mission within the labs, we have discussed the topic during the recitations and provided possible algorithms for this mission. At the third stage, the robot is expected to detect the first red cylinder and follow the wall in the counterclockwise (CCW) direction until it sees the next red cylinder. Then, it should repeat the steps for the second one. After following the second red cylinder wall in CCW, the robot should reach the given goal position. The competition environment is shown in Fig. 6. We provided the students with virtual and actual environments for two weeks during recitation and lab sessions before the competition to build the map and improve their algorithms. Due to the social distancing, only 30 minutes were given to each team for the competition. While all teams could accomplish at least one stage in the virtual environment, only a few teams could complete in the real environment. However, we had at least one team within ten teams for each mission that completed perfectly.

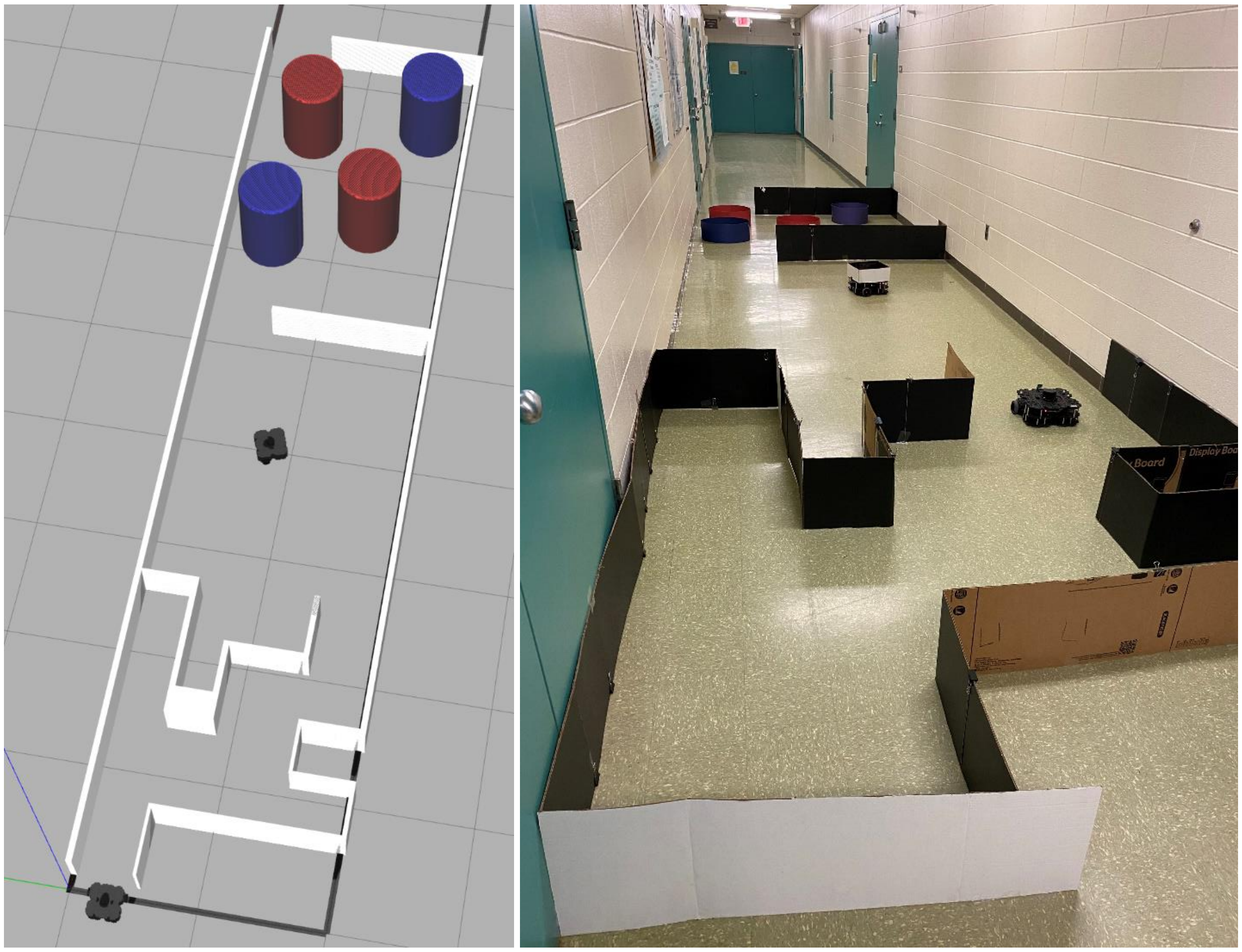

Fig. 6. The competition environments in simulation (left) and real world (right). Stage 1 is a maze-like environment; Stage 2 is an environment with dynamic obstacle (another robot here) and Stage 3 is an environment with static obstacles.

One team implemented a state-of-art algorithm introduced during the recitation for stage 2, dynamic obstacle avoidance. As the course was designed for seniors and graduate students, most of the students aimed to pursue their careers in robotics. They were motivated to work on improving their algorithm performance as they were already interested in the topics. Having the system dynamics course listed as a prerequisite course, some students wanted to design their own PID trajectory tracking controller instead of time-based control. We plan to include this as an optional lab activity.

Successes and Challenges

First of all, having Windows-based PCs in the lab caused some issues at the beginning of the semester. Although ROS started with a Linux-based operating system, we had to work with Windows-based PCs as both the lab and most students' PCs were Windows. Unfortunately, some students struggled with the installation due to the instability of the software on the Windows platform. For example, two students who had space on their user names could not install ROS as there is space on their folder names and had to lease another laptop. In addition, while most of the course utilized software is available in the lab computers through AppsAnywhere [3], it was not available to do the same for ROS. As the Linux-based ROS is much more stable, we used the Linux-based laptops for operating robots, which caused another step for the students to learn to launch their implemented file due to the system differences. If any lab environment with Linux-based PCs is available, the course preparation will be much more manageable.

Secondly, as the department did not have a proper prerequisite course for basic programming, students were not familiar with programming at a comfortable level. Until Lab 4, students struggled so much with coding, even if the template codes and reference pages were provided with additional explanations before the lab sessions. Therefore, any programming-related course, even if it is not Python, is recommended as a prerequisite to the course. However, most of the students appreciated that they became comfortable with ROS and Python at the end of the semester, showing that our learning objective was successfully achieved.

While trying to reduce the pressures on students by not mandating the results on simulation environments for their pre-lab reports but still requiring them to discuss their plans, we observed that some students open the provided material for the first time during the lab sessions. This inadequate preparation caused delays in their post-lab reports, as they usually needed additional lab time to finish their missions. Thus, we plan to mandate the students to submit the implementation results while giving some relaxed rubric on the grading.

Lastly, while implementation in the simulation environments helped students explore before working in real environments, sometimes it gave the students double loads on their assignments. They had to do the same thing twice, especially when hardware factors were involved in the missions. However, by experiencing both sides, students realized the difficulties of working with real robots, which is advantageous.

Conclusion

This paper presented a course designed to introduce autonomous navigation focusing on hands-on experiments. While the lab sessions deal primarily with real robots, students learn core algorithms required for autonomous navigation throughout the course, which is the software side of robotics. We observed that providing experiences in simulation and real environments was effective while giving students the freedom not to present to the lab multiple times a week. We plan to further improve the course by revising pre-lab assignments and resources to enhance programming skills. We also plan to collaborate with other instructors to provide an adequate prerequisite course for Python programming.

References

1. “The robot operating system (ROS), 2021. Accessed August 8, 2021. [Online]. Available: https://www.ros.org/.
2. “Turtlebot3 waffle pi,” 2022.Accessed August 8, 2021). [Online]. Available: https://www.robotis.us/turtlebot-3-waffle-pi/.
3. “Appsanywhere,” 2021. Accessed August 8, 2021). [Online]. Available: https://www.software2.com/appsanywhere/what-is-appsanywhere.

Biographies

ABHISHEK PATIL is currently pursuing a PhD in mechanical engineering at Michigan Technological University, Houghton, MI, USA. He received his BE and MS in mechanical engineering from Punjab Engineering College, Chandigarh, India, in 2011 and from Michigan Technological University in 2020, respectively. He has worked as a brake system design and development engineer at Maruti Suzuki India Ltd, India. He has worked as a teaching assistant and developed ROS-based lab environments for an undergraduate course on autonomous systems. His primary interest is developing and experimenting with optimization algorithms for robotic applications.

JUNGYUN BAE is an assistant professor in the Departments of Mechanical Engineering-Engineering Mechanics and Applied Computing at Michigan Technological University, Houghton, MI. She received her BS and MS in mechanical engineering at Hongik University, Seoul, South Korea, in 2005 and 2007 respectively, and PhD in mechanical engineering at Texas A&M University in College Station, TX, in 2014. Prior to joining Michigan Tech, she was a research professor in the Intelligent Systems and Robotics Laboratory at Korea University, Seoul, South Korea. Her research interests include multi-robot system optimization and autonomous navigation.