ASEE 2022 ANNUAL CONFERENCE
Excellence Through Diversity
MINNEAPOLIS, MINNESOTA, JUNE 26TH-29TH, 2022 ASEE
Paper ID #37445

A project-based platform for students' Robot Operation System (ROS) programming experience

Yifan Wang

Zhou Zhang (Dr.)

Yizhe Chang

Yizhe Chang is an assistant professor in mechanical engineering.

2022 ASEE Annual Conference & Exposition
Minneapolis, Minnesota, USA, June 26-29, 2022
Wang, Y., Zhang, Z., Chang, Y.

A project-based platform for students' Robot Operation System (ROS) programming experience

Yifan Wang, Zhou Zhang, Yizhe Chang

Abstract

Robot Operating System (ROS) is an open-source software framework for robot automation. ROS includes a protocol of robot commanding and integrates with a collection of software libraries. ROS aims to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. Currently, ROS is the major robotics software framework for researchers in graduate schools.

One of the obstacles in exposing ROS to students is lacking appropriate projects for starters. ROS is not learner-friendly for its steep learning curve and comprehensive programming background needed. In this article, a robotics platform is designed by the author's institution, based on an iRobot Create robot, is introduced. The robotics platform that includes a mobile base, a webcam, and a Lidar, can fully support ROS-based programming for autonomous navigation. To help students start ROS programming practicing, a ROS-based student autonomous navigation project, including components of path-planning, image recognition, and simultaneous localization and mapping (SLAM) is implemented on this robotics platform. A pilot study shows that the platform is favored by students and inspiring for students to learn ROS. However, the curriculum needs improvements to ensure the students' success in completing the tasks.

Introduction

With the advancement of sensing and control technology, lower price in computation units and sensors. The applications of robots have been extended broadly from their traditional industry roles to all aspects of our daily activities, such as medical, military, transportation, companion, as well as more recently, household service

Robotics education has been attracting increasing attention. Educators have been exploring the methods of robotics education for all ages [1]. Traditional robotics education heavily focuses on mechanical design, circuit design, sensor application, and control theories [2]. Recent developments in robotics pose new challenges for educators. How can we integrate breakthroughs in artificial intelligence, computer vision, sensor fusing, into the classroom, for students to experience? On one hand, research finds project-based learning can be effective [3] [4]. Through projects (and student competitions), students can get into the applications of a new domain, without the need for extensive theory learning and lengthy programming. On the other hand, new robotics software platforms bring new opportunities. Among these platforms, MATLAB and Robot operation system (ROS) are the most popular ones.

MATLAB, a proprietary multi-paradigm programming language, and numeric computing environment, gained many popularities [5] [6] [7]. However, the proprietary status, the limited processing speed, as well as MATLAB-specific *.m* programming language, slowed MATLAB further application in the industry.

Robot operation system (ROS), an open-source platform for robot programming, is becoming popular [8] [9]. Running on Ubuntu Linux (free of cost for users), with support of both general-use programming language: Python and C/C++, allow a bigger community to contribute to the robotics society. More importantly, ROS adopts a framework configuration that is widely accepted by robot development communities [10] [11] [12]. Existing ROS-based robotics curriculum shows positive feedbacks from their learners [13] [14] [15].

In this paper, we introduced an educational robot platform. With hobby-level components, the robotics platform that includes a mobile base, a webcam, and a Lidar, can fully support ROS-based programming for autonomous navigation. The platform, in combination of hardware and software, can not only enhance students' theory understanding, but also provide ROS programming experience on state-of-the-art robotics topics such as object detection, LiDAR sensing, path planning, robot dynamics, simultaneous localization and mapping (SLAM) etc.

A case study using this platform is introduced in the paper. In the case study, students finished a house-hold robot project "toy car pick and removal" using the platform. In the project, students gained hands-on experienced on all topics of autonomous navigation.

Brief introduction to Robot Operation System (ROS)

ROS is an open-sourced and distributed middleware framework for developing applications of robots, ROS is officially operated on Linux operating systems. There are over 2,000 software packages, written and maintained by almost 600 people. Approximately 80 commercially available robots are supported. The primary goal of ROS is to secure modularity and reusability in robotic research and development. Therefore, the ROS is organized as packages, where minor changes are required before application or integration.

The center of ROS is the modularity. A strict structure of robot operating, consists of ROS node, topic, message, service, etc., is defined. A node is an executable file that performs computation and communication. The information passed between nodes is called messages. A topic is used for continuously publishing or subscribing messages unidirectionally. Service is another way to pass data between nodes. For the tasks occasionally executed within a bounded period, services allow nodes to request messages and await responses synchronously. Nodes can publish or subscribe to messages through a topic or exchange a synchronous request-and-response message through service. Action is an advanced interface to implement goal-orientated, time-extended behaviors. While services are synchronous, actions are asynchronous. An action initializes a behavior with services for goal and result. It also provides a topic for feedback and allows the goal to be canceled. The communication mechanism between nodes with a topic, a service, or an action at the computational graphic level is shown in Figure 1. ROS also provides an effective

visualization tool, RViz (the acronym for ROS visualization), which permits users to see robot models, sensor data, or other visualizable topics.

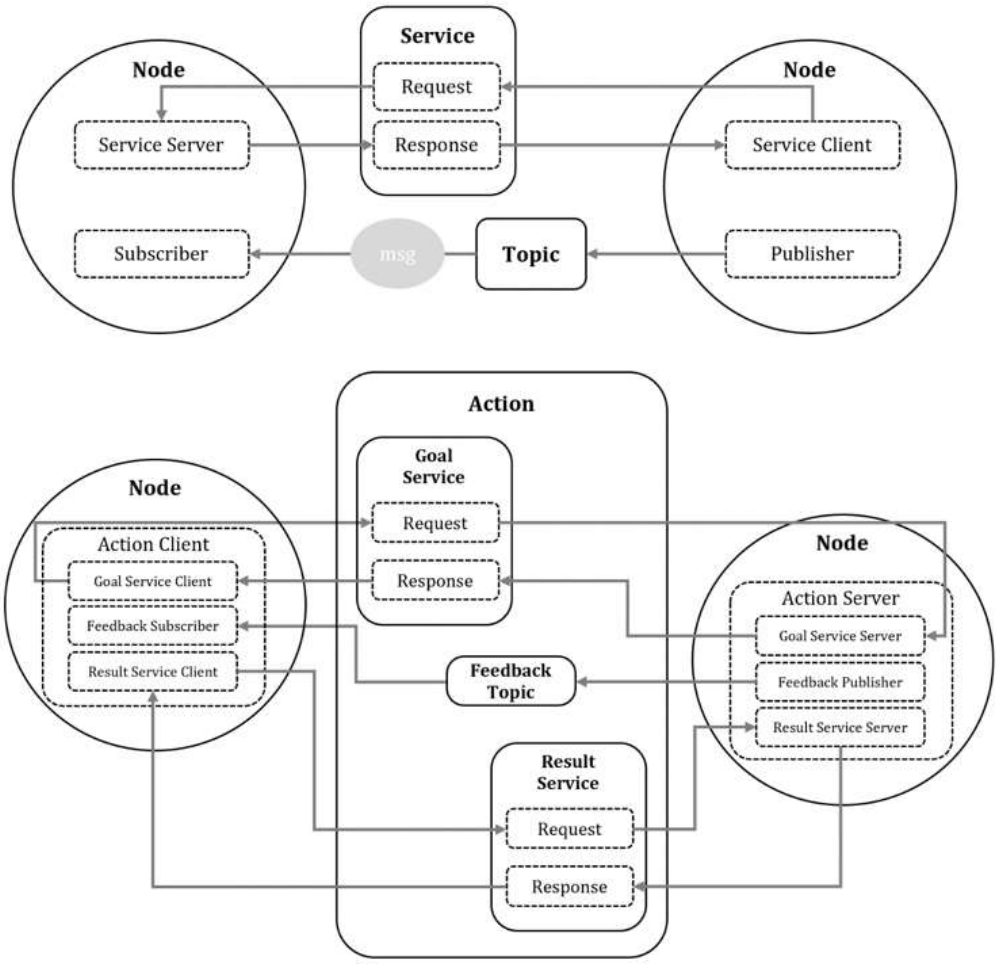

Figure 1 Structure of ROS

Robot platform hardware

The robot platform includes a mobile base, a 3d printed dock and accessories, a LiDAR sensor, and two cameras (see Figure 2).

iRobot Create 2 is an economical and programmable differential drive robot. Differential wheeled mobile robots can be separately driven based on linear and angular velocities. By employing a differential drive robot, the geometric constraints can be disregarded while controlling the robot to move along a planned path. Considering the maneuverability and economic efficiency, it is chosen as the moving base for the platform.

A mini gripper using 3d printing is implemented to minimize the cost. It is not applicable to grip the detected objects with the gripper on the robotic arm. A solution to place the toy car in a designated position is to dock the detected toy car and keep it ahead of the robot during moving

and rotating until it reaches the goal. The length and width are tested to guarantee its ability to keep objects inside while moving forward and rotating and to let small objects escape from it while moving backward. The robot can be controlled after checking the relative position between the dock and the detected objects.

A LiDAR sensor is integrated for SLAM and obstacle detection during navigation. Considering the application environment and economic efficiency, SLAMTEC RPLidar A1M8, is used. This 2D LiDAR sensor can measure a range of 0.15 – 12m, for 360 degrees. Compared to many commercial LiDAR sensors, the cost of SLAMTEC RPLidar is low.

Two low-cost cameras are applied – a close-up camera and a full-shot camera, the close-up camera, is for checking the relative position between the objects and the dock and controlling the robot accurately to place the objects inside the dock.

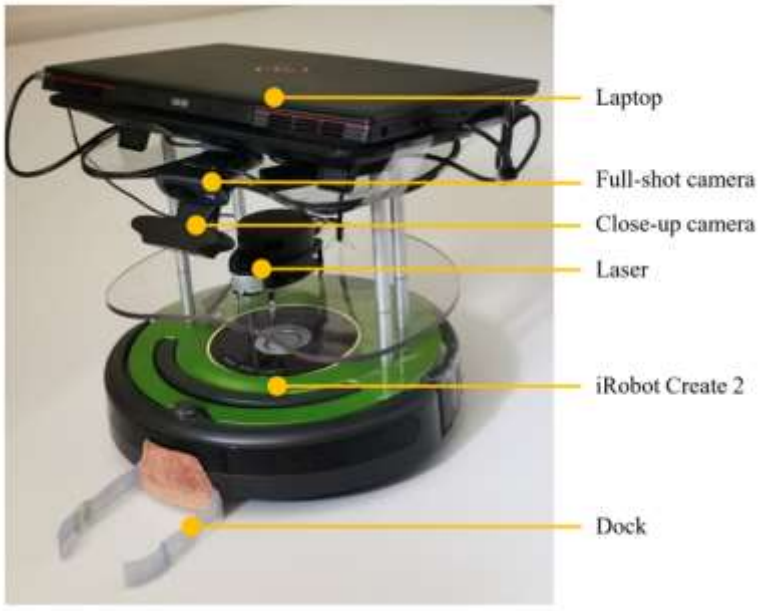

Figure 2 The robot platform. The laptop can be replaced by a Raspberry Pi for reduced cost

Object Detection module

One of the experiences for students is visual-based object recognition. Based on an existing ROS-integrated visual object recognition package ("YOLO-You Only Look Once"), our platform allows students to integrate state-of-the-art visual object recognition technology.

The "You Only Look Once" (YOLO) package is a real-time object recognition based on convolutional neural network. The first version was published by Joseph Redmon in 2016 [16] [17] [18]. "YOLO" network allows multiple object recognition at high accuracy [19]. Multiple versions of YOLO networks are implemented, including version 4 (YOLOv4) on the darknet platform, and version 5 (YOLOv5), which is integrated into ROS [20].

YOLOv5 contains four types of architectures which are named with suffix s for small, m for medium, l for large, and x for extra-large, according to the number of residual units in CSP1_X, CBL in CSP2_X, and convolutional kernel number.

Simultaneously localization and mapping (SLAM) module

Augmented Monte Carlo Localization (AMCL) is used for students' SLAM experience. The AMCL package implements the localization function based on probability. It is an improvement of basic Monte Carlo Localization (MCL).

The MCL is aimed to match the laser data and the grid occupancy map and to predict the localization in the grid occupancy map with the highest probability based on the laser data (shown in Figure 3).

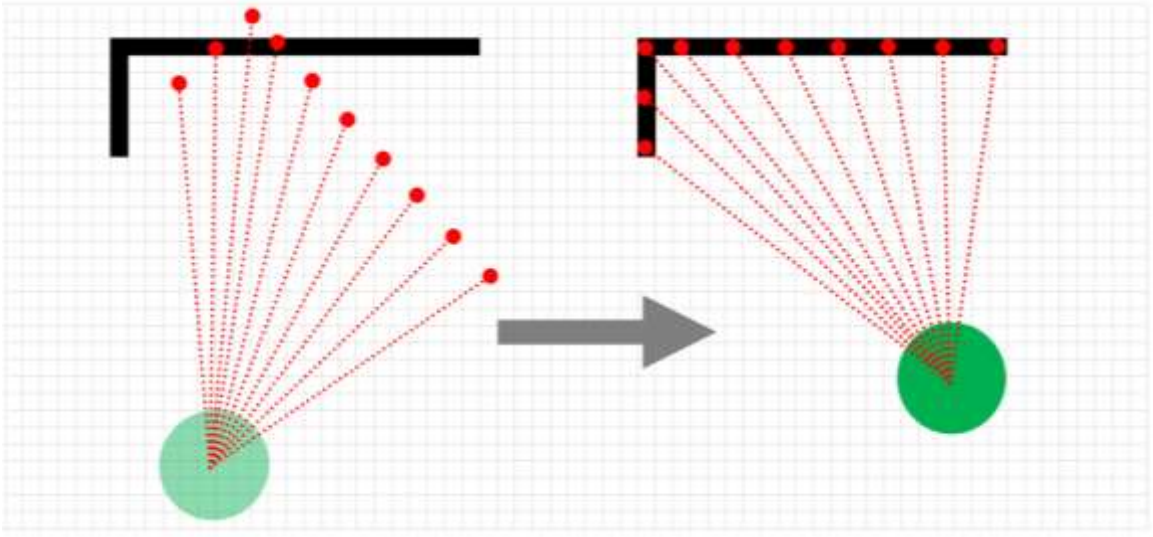

Figure 3: The Illustration of MCL localization

The pseudocode of basic MCL is shown in Table 1. Basic MCL will first randomly generate a bunch of particles $X_t = \{x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]}\}$ and predict the movement of the particles based on the odometry motion model. Then, update the weights of each particle based on the beam measurement model and replace the low-weighted particles with copies of high-weighted particles. Then compute the weighted mean and covariance of the resampled particle set to obtain the estimation of the state of the robot.

Table 1: Pseudocode for Basic MCL Algorithm [21]

Algorithm MCL(X_{t-1}, u_t, z_t, m)**:**
$\overline{X_t} = X_t = \Phi$;
for $m = 1$ *to M do*
$x_t^{[m]} = sample\_motion\_model(u_t, x_{t-1}^{[m]})$
$w_t^{[m]} = measurement\_model(u_t, x_t^{[m]}, m)$

$\overline{X_t} = \overline{X_t} + \left(x_t^{[m]}, w_t^{[m]}\right)$
endfor
for $m = 1$ *to* M do
draw $i \in \{1, \ldots, N\}$ *with probability* $\propto w_t^{[i]}$
add $x_t^{[i]}$ *to* X_t
endfor
return X_t

Map construction module

For mapping, GMapping, a software package based on LiDAR sensing is used. So far, Gmapping is considered one of the most effective software packages in ROS community [22].

After a map is constructed by GMapping, *map_server* needs to be executed to save the map for future usage. The map will be saved as an image of the grid-occupancy map and a configuration file in *yaml* format describing the parameters of the map.

The Gmapping package requires the frame transformation from the LIDAR to the robot (published by *robot_state_publisher* or *static_transform_publisher*), and from the robot base to the robot odometry (usually published by odometers) and then publishes that from the map to the odometry. The ROS computation graph of Gmapping nodes is shown in Figure 4.

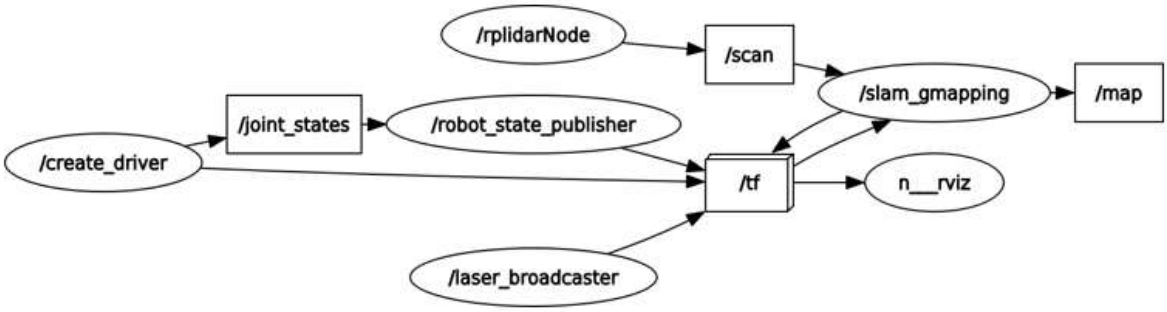

Figure 4 GMapping architecture in ROS

Path planning and locomotion module

Dijkstra's algorithm is a graph-based motion planning and was published by E. W. Dijkstra in 1959 [23]. It works by visiting nodes layer by layer. Beginning from the start. it then repeatedly explores the unvisited adjacent nodes of the node visited, expanding outwards from the start to the goal. Dijkstra's algorithm guarantees to find the shortest path. In the projects, students can use a software package “move-base” for learning path planning.

The locomotion module, also referred to as the "local planner" in autonomous navigation, works with the dynamic states of the robot currently obtained from the sensors and only plans the trajectories within the vicinity of the robot. Local planners also publish velocity commands frequently to drive robots based on local plans. The ROS navigation stack provides a local planner using the trajectory rollout algorithm (*base_local_planner*), as well as a local planner using the dynamic window approach (*dwa_local_planner*). ROS also allows users to apply other customized local planners, such as the one using the follow the carrot algorithm (*asr_ftc_local_planner*).

Using RViz, the elements of robot autonomous navigation can be visualized for students debugging in Figure 5.

Project case-study: household service robot for removing toy cars scattered on the floor

One of the applications of this platform is a project on a household service robot. Toy cars are on most children's favorite toy lists and many children may own a bunch of toy cars. Similar scenarios may appear in every family with children in which a child stops to do something else, for example, sleep or eat, after playing with toy cars without putting them away. The forgotten toy cars scattered on the floor impose danger to all members of the house.

Using the robot platform, students can program an autonomous robot to detect and locate the scattered toy cars, plan a path to autonomously navigate to a target toy car, "grip" the toy car, navigate to the toy car collection area. Then repeat for another detected toy car. Figure 6 shows the toy car removal flowchart.

The robot was executed in the scenario shown in Figure 7. The yellow dashed line is the expected patrol routine, and the green arrowed line indicates the designated poses. The final goal is the red arrowed line in the bottom right.

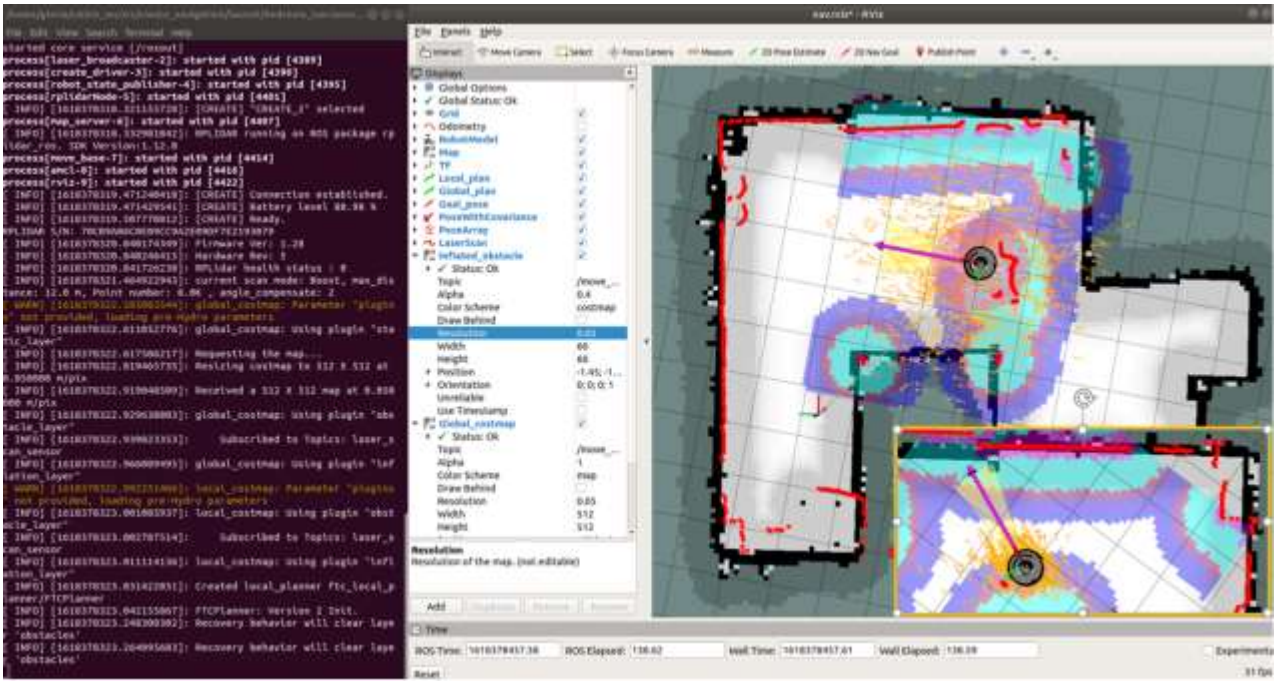

Figure 5 Visualization of autonomous navigation using RViz

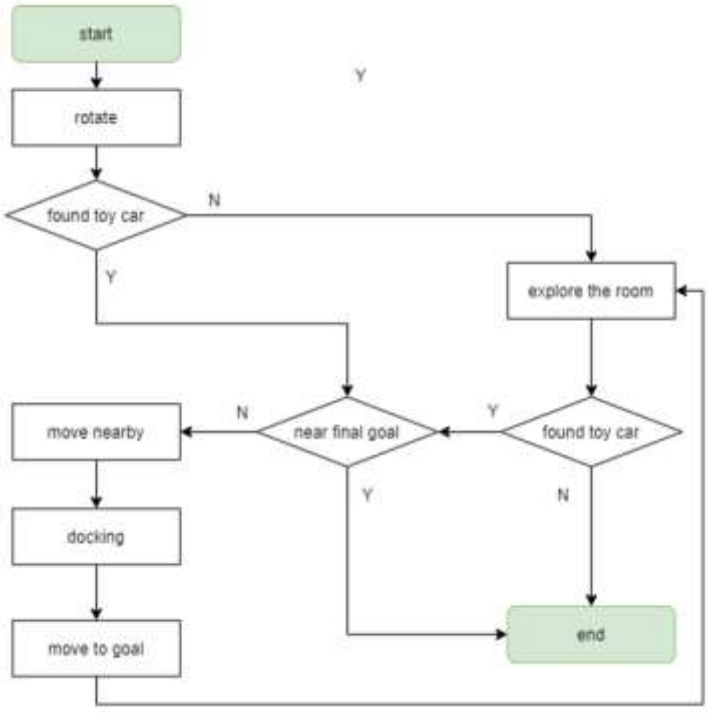

Figure 6 Flowchart for toy car detection

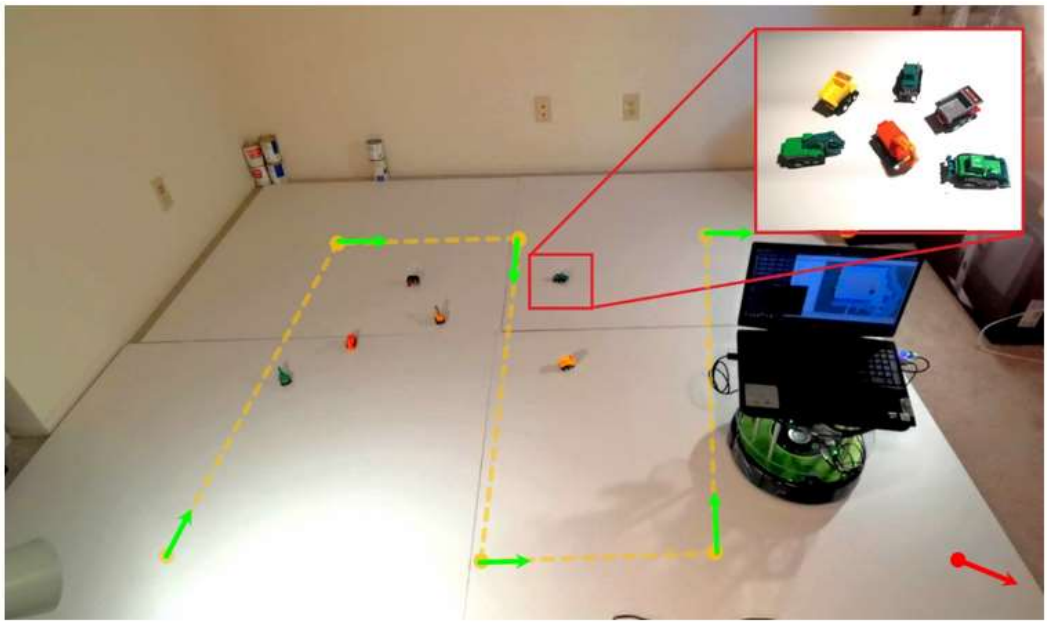

Figure 7 A scenario for autonomous toy car removal

To evaluate the outcome of the toy car removal project, various test runs were conducted. Each run was evaluated individually by its total time, the average time to place one toy car, accuracy (equals to the ratio of the number of the detected toy cars to the total number of toy cars on the ground), and success rate (equals to the ratio of the number of the successfully placed toy cars to the total number of toy cars on the ground). The average result indicates the general performance of the system.

Pilot study on students' learning using the platform

As a pilot test, 3 senior-level mechanical engineering undergraduate students were invited to test the platform using the service robot scenario. These students were selected because they had some previous ROS programming background.

The background survey result of the students' knowledge of robotics and programming is shown in Table 2.

Table 2 Student's background in robotics

Student	Robotics hardware experience	Previous Experience in Programming (any language)	Previous experience in Python language	Experience in ROS before
#1	>2 years	>2 years	>2 years	<3 months
#2	>2 years	>2 years	1-2 years	<3 months

#3	1-2 years	>2 years	1-2 years	<3 months

From the background, most of them have previous robotics experience and programming experience. However, all of them started to learn ROS only recently for their senior design projects.

The students were provided with all modules (object detection module, SLAM module, Map construction module, and Path planning module) with instructions. They were given 3 hours to "assemble" these modules to complete the flowchart in Figure 6 to complete the toy removal projects. In the task, they need to correctly identify the "*topics*" needed and provided by the modules; then, subscribe to or publish the *topics* accordingly to complete a ROS node in Python for the toy car identification-and-removing task.

Only one student (student #1) could finish the task and complete 10-time test runs. Since the system requires the final check during one patrol circle, the difference between the total time and the time for one patrol circle is divided by the number of placed toy cars to compute the average time for placing one toy car. The test-run results are shown in Table 3, the average time to place one toy car is about one minute. The average accuracy of real-time detection is 83.1% and the success rate of placing the toy car at the designated final goal is 78.9%.

The other two students were unable to complete the ROS node in Python.

Table 3: The Performances for the Detection and Placing of Different Numbers of Toy Cars

Run #	Total Time (s)	Toy car #	Detected #	Placed #	Accuracy (%)	Success Rate (%)	Avg. time per toy car (s)
1	183	1	1	1	100.0%	100.0%	79
2	190	2	1	2	50.0%	100.0%	43
3	212	2	2	2	100.0%	100.0%	54
4	252	3	2	2	66.7%	66.7%	74
5	303	3	3	3	100.0%	100.0%	66
6	-	3	3	1	100.0%	33.3%	-
7	350	5	4	4	80.0%	80.0%	62
8	320	5	5	5	100.0%	100.0%	43

9	394	6	5	5	83.3%	83.3%	58
10	232	6	3	3	50.0%	50.0%	43
11	-	6	4	2	66.7%	33.3%	-
12	475	6	6	6	100.0%	100.0%	62
Average					**83.1%**	**78.9%**	**60**

Discussion

A post-interview was given to all students. The students need to respond to the following questions using the Likert scale (1-strongly disagree, 2-disagree, 3-neutral, 4-agree, 5-strongly agree):

1. The platform hardware and software modules meet the state-of-the-art robotics application scenarios
2. The platform can be used for different senior-design robotics projects
3. It is easy to start using the platform
4. I would like to recommend the platform to my classmates

Table 4 Post-interview results

Student	Q1	Q2	Q3	Q4
#1	5	4	4	5
#2	5	4	3	4
#3	4	4	2	4
Avg	**4.67**	**4**	**3**	**4.33**

The students all agree that the platform provides the possibility to practice new concepts of object detection, SLAM, mapping, and path planning. As these concepts are new and mostly not covered in any undergraduate curriculum, it is exciting to be learned in a senior design. They also believe the platform can be used in various senior design projects for different applications.

Two students who could not finish the task felt that the main obstacles were the evaluation process. The sharp learning curve made it hard to complete the tasks in time. The students suggested instead of one 3-hour session of learning and evaluation, there should be a series of

laboratory exercises to help students to get into the project. Nonetheless, they both claimed that they look forward to getting into the state-of-the-art topics that the platform can introduce

Conclusion and discussion

In this paper, a platform for students' Robot Operation System (ROS) programming experience is designed and evaluated. Using hobby-level hardware: an iRobot create 2 mobile base, webcams, 2D LiDAR, and 3d printed accessories, students who are not in robotics concentration can experience project-based ROS programming. The adapted software packages, in combination with the platform hardware, projects for autonomous navigation can be achieved by students.

As a case study, a household service robot application is achieved using the platform. Students programmed using ROS to experience the whole process of autonomous navigation: object detection, path planning, localization, mapping, and robot locomotion. The case study finds a real-world scenario on toy car removal, which can reduce the workload of parents and increase home safety.

So far, the projects on the platform are not integrated into any established curriculum. The pilot test showed students may be able to complete a relatively complicated task with ROS background. The pilot test also showed that a well-designed laboratory series is needed to boost the success rate of students. Thus, with further training, students can use the platform for independent projects (senior design, master's project, etc.). Using this platform with improved training and exercising session, students will gain hands-on experience in implementing all components of robot autonomous navigation and overcoming the steep learning curve of ROS.

Bibliography

[1] M. J. Mataric, "Robotics education for all ages," in *Proc. AAAI Spring Symposium on Accessible, Hands-on AI and Robotics Education*, 2004.

[2] R. Manseur, "Development of an undergraduate robotics course," in *Proceedings Frontiers in Education 1997 27th Annual Conference*, 1997.

[3] B. A. Maxwell and L. A. Meeden, "Integrating robotics research with undergraduate education," *IEEE Intelligent systems and their applications* , vol. 15, no. 6, pp. 22-27, 2000.

[4] I. R. Nourbakhsh, K. Crowley, A. Bhave, E. Hamner, T. Hsiu, A. Perez-Bergquist, S. Richards and K. Wilkinson, "The robotic autonomy mobile robotics course: Robot design, curriculum design and educational assessment," *Autonomous Robots*, vol. 18, no. 1, pp. 103-127, 2005.

[5] J. M. Esposito, "The state of robotics education: Proposed goals for positively transforming robotics education at postsecondary institutions," *IEEE Robotics & Automation Magazine* , vol. 24, no. 3, pp. 157-164, 2017.

[6] A. Gil, O. Reinoso, J. M. Marin, L. Paya and J. Ruiz, "Development and deployment of a new robotics toolbox for education," *Computer Applications in Engineering Education* , vol. 23, no. 3, pp. 443-454, 2015.

[7] R. Gonzalez, C. Mahulea and M. Kloetzer, "A Matlab-based interactive simulator for mobile robotics," in *IEEE International Conference on Automation Science and Engineering (CASE)*, 2015.

[8] J. M. Canas, E. Perdices, L. Garcia-Perez and J. Fernandez-Conde, "A ROS-based open tool for intelligent robotics education," *Applied Sciences*, vol. 10, no. 21, p. 7419, 2020.

[9] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, Kobe, Japan, 2009.

[10] M. Cooney, C. Yang, A. Padi Siva, S. Arunesh and J. David, "Teaching robotics with robot operating system (ros): A behavior model perspective," in *Workshop on "Teaching Robotics with ROS", European Robotics Forum* , Tampere, Finland, 2018.

[11] G. Sidorenko, W. Mostowski, A. Vinel, J. Sjoberg and M. Cooney, "The CAR Approach: Creative Applied Research Experiences for Master's Students in Autonomous Platooning," in *IEEE International Conference on Robot & Human Interactive Communication (Ro-man)*, 2021.

[12] K. Alisher, K. Alexander and B. Alexandr, "Control of the mobile robots with ROS in robotics courses," *Procedia Engineering*, vol. 100, pp. 1475-1484, 2015.

[13] S. a. G. S. Michieletto, E. Pagello, M. Moro and E. Menegatti, "Why teach robotics using ROS," *Journal of Automation, Mobile Robotics and Intelligent Systems*, pp. 60-68, 2014.

[14] N. F. Ferreira, A. Araujo, M. Couceiro and D. Portugal, "Intensive summer course in robotics--Robotcraft," *Applied Computing and Informatics*, 2018.

[15] M. Kandlhofer, G. Steinbauer, M. Menzinger, R. Halatschek, F. Kemeny and K. Landerl, "MINT-Robo: Empowering Gifted High School Students with Robotics," in *IEEE Frontiers in Education Conference (FIE)*, 2019.

[16] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, 2016.

[17] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, 2017.

[18] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018.

[19] A. Bochkovskiy, . C.-Y. Wang and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," 2020.

[20] G. Jocher, "YOLOv5," [Online]. Available: https://github.com/ultralytics/yolov5.

[21] S. Thrun, Probabilistic Robotics, Cambridge, Massachusetts: The MIT Press, 2006, p. 245.

[22] X. Zhang, J. Lai, D. Xu, H. Li and M. Fu, "2D Lidar-Based SLAM and Path Planning for Indoor Rescue Using Mobile Robots," *Journal of Advanced Transportation,* 2020.

[23] E. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik,* vol. 1, pp. 269-271, 1959.

[24] Open Robotics, "ROS Introduction," [Online]. Available: http://wiki.ros.org/ROS/Introduction.

[25] M. Quigley, B. Gerkey and W. D. Smart, Programming Robots with ROS: A Practical Introduction to the Robot Operating System, O'Reilly Media, Inc, 2015.

[26] Open Robotics, "ROS Concepts," 21 June 2014. [Online]. Available: http://wiki.ros.org/ROS/Concepts.

[27] T. Shah, "ROS Basics 1 - Nodes, Topics, Services & Actions," 1 April 2017. [Online]. Available: https://tarangshah.com/blog/2017-04-01/ros-basics-1-nodes-topics-services-actions/.

[28] SLAMTEC, "Specifications of RPLidar A1M8," [Online]. Available: http://www.slamtec.com/en/Lidar/A1Spec.

[29] D. Jiang, "An Overview of YOLO," 2020. [Online]. Available: https://zhuanlan.zhihu.com/p/143747206.

[30] J. Nelson and J. Solawetz, "YOLOv5 is Here: State-of-the-Art Object Detection at 140 FPS," 10 June 2020. [Online]. Available: https://blog.roboflow.com/yolov5-is-here/.