Implementing and Using ROS in Undergraduate Robotics Curricula

Prof. Siobhan Rigby Oca, Duke University

Siobhan Rigby Oca is an Assistant Professor of the Practice and Assistant Director of Robotics Programs in the Thomas Lord Department of Mechanical Engineering and Materials Science at Duke University, NC, USA. She received her B.Sc. from Massachusetts Institute of Technology and Masters in Translational Medicine from the Universities of California Berkeley and San Francisco. She completed her Ph.D. in Mechanical Engineering at Duke University. Her research interests include applied medical robotics, human robot interaction, and robotics education.

Dr. Blake Hament, Elon University

Blake Hament is an Assistant Professor of Engineering at Elon University. The work presented in this manuscript was completed during his Ph.D. program at University of Nevada, Las Vegas. Blake was awarded a US Congressional Commendation and a US Department of Transportation Outstanding Student of the Year award for his contributions. He received a B.S. in Physics from Duke University and served as a research assistant at the European Center for Nuclear Research (CERN). After his undergraduate studies, Blake joined Teach for America, served as a robotics coach, and earned his M.Ed. in Science Education from University of Nevada, Las Vegas. He earned his Ph.D. with the Mechanical Engineering Department at University of Nevada, Las Vegas while conducting R&D with companies like Tesla, Lockheed Martin, Boston Dynamics, and local aerospace and robotics startups.

Implementing and Using ROS in Undergraduate Robotics Curricula

1 Abstract

This review aims to elucidate multiple options, challenges, and opportunities to incorporate ROS into undergraduate robotics courses. First, the importance of ROS in the robotics research and industry community is discussed as motivation to learn how to use this middleware (and frequently used packages) in the classroom. Additionally, examples of use in the classroom and challenges of implementation are described based on both literature and interviews with instructors across the country. Then, specific implementation approaches for getting students started using ROS are introduced/described along with specific examples and pros/cons of each approach. Finally, ROS1 versus ROS2 is discussed to describe the utility of each option for instructors as they develop their courses. Overall, this review is meant to collate motivation and options for instructors in robotics trying to incorporate ROS into their courses with minimal overhead for themselves and students.

2 Introduction

According to the 2022 ROS Metrics Report, ROS has been cited in academic papers 10,467 times and more than 740 companies are actively using ROS as a tool [1]. These metrics are growing steadily year over year. While early adopters of ROS were graduate students or industry users, increasingly, students and instructors are taking an interest in ROS at the undergraduate level [2] [3] [4] as has been explored at the masters level [5] [6]. However, even just installing ROS can be a daunting task for the uninitiated. This paper explores options for installing ROS for undergraduate courses, offers recommendations, and points readers towards additional guides and resources.

2.1 Importance of ROS

ROS has become a powerful staple of robotics research and development. ROS is a software suite with efficient, modular, and easily customizable software tools [7]. It is free to use, well-documented, and widely supported. Robotics researchers and developers can quickly

spin up projects using ROS packages, devoting their time to novel robotic applications rather than reinventing the "wheel" of tried-and-true low-level software programs for communication, visualization, and resource management [8] [9]. ROS was created to be the "Linux of Robotics," and to this day it is supported by an international community of open-source contributors.

ROS has long straddled the academic and industrial research communities. It began as an ambitious project by Keenan Wyrobek and Eric Berger at the beginning of their PhD's at Stanford. Development skyrocketed when the project moved to Scott Hassan's Willow Garage technology incubator in concert with the development of the PR1 robot line. Today, ROS can be found in virtually every academic robotics research lab, and industry leaders like Boston Dynamics and iRobot use ROS to prototype and test their products [7]. Thus, students who learn ROS gain an important and ubiquitous skill that sets them up for success in academia and industry.

2.2 Function of ROS in Undergraduate Courses

ROS has been used in undergraduate robotics courses as a tool to learn about, simulate, and actuate robotic mechanisms[10, 11]. On top of ROS, there are many packages that can be used to visualize physical environments, like Gazebo, and develop, simulate and execute motion plans, like MoveIt, for examples. Examples of implementation include course projects, some of which use ROS for connecting hardware in applied projects. Additionally, ROS can be a main subject of courses at the undergraduate and graduate level for its utility in industry and example for learning about middleware in general, like understanding communication or sequencing [12].

2.3 Challenges of using ROS in Undergraduate Curricula

Though learning how to use ROS has a significant learning curve, it is considered foundational for many applied roboticists and is often found in robotics job listings. Key limitations in using ROS include the learning curve and the computational resources needed to run the program[13]. ROS was designed to run on Linux based systems, although options are described in detail in this paper. Streamlining these processes for instructors and students and making sure students have access to sufficient computational power to run simulations is a huge barrier for teaching and implementing ROS.

3 Review of ROS Implementation Approaches

There are several different options for running ROS regarding operating systems (OS). These options are discussed in-depth below and illustrated in Fig.1 and Fig.2.

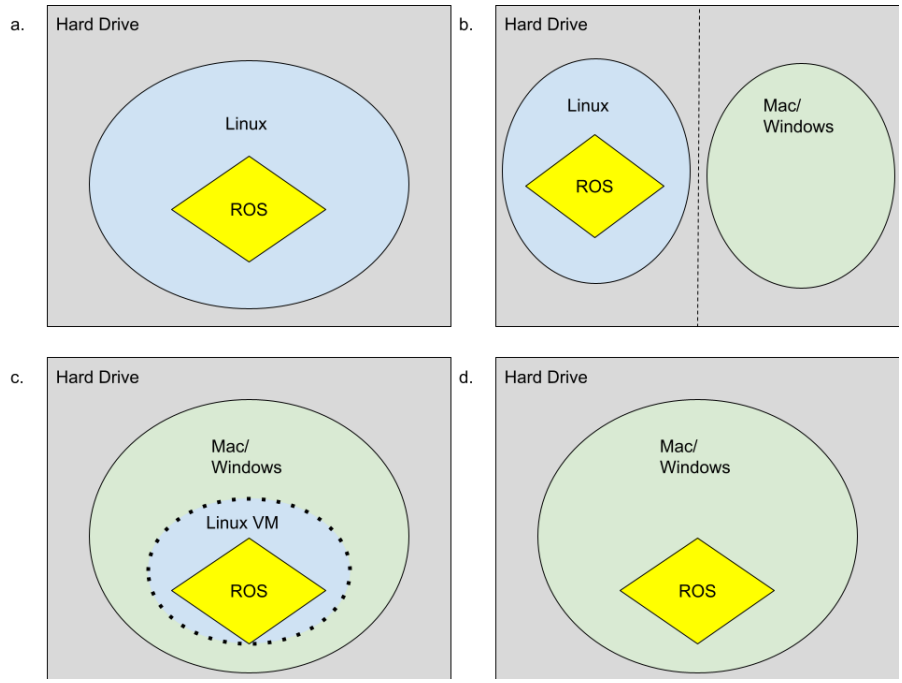

Figure 1: The various configurations for ROS installation include: a. directly installing Linux, b. dual-booting with Linux and Windows or Mac, c. running ROS via a Linux VM on a Windows or Mac machine, or d. running the new ROS for Mac or ROS for Windows packages directly on the appropriate OS.

.

3.1 Linux via Dual Boot / Local Installation

3.1.1 Overview of Implementation

ROS1 is officially recommended for use with the Ubuntu and Debian Linux OS. These OS can be directly installed to a machine in place of Windows or Mac OS or installed alongside them with dual-boot configuration. Directly installing a Linux OS and ROS requires the user to reserve a partition or all of their hard drive to run the software. Dual-booting allows the user to retain their original OS while installing the new OS on the same hard drive. The memory allocated to each OS is only accessible when that specific OS is being run.

There are several pros and cons to directly installing, whether as the sole OS or via dual-boot.

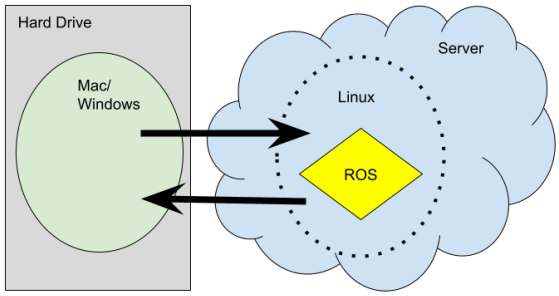

Figure 2: Another possible configuration for ROS includes connecting to Linux instances hosted on remote servers as shown above.
.

Users who are unfamiliar with installing OS and partitioning memory run the risk of corrupting some or all of their hard drives and losing data. However, there are numerous official guides and user posted tutorials that demonstrate how to navigate this process and directly install Linux OS by creating bootable USB sticks or startup discs. One of the major benefits of directly installing a Linux OS and ROS is that most hardware communication and peripherals will work off-the-shelf. This can be especially important when interfacing with actuators, cameras, and other sensors over USB and Serial Ports.

3.1.2 Example Implementation

The Université de Sherbrooke offers a Bachelors of Engineering with concentration in Robotics. They maintain a computer lab with Linux installed directly on all machines, specifically for ROS development. They have found that many students additionally install ROS on their personal laptops using virtual machines (VM), discussed in greater detail below [11].

3.1.3 Recommendations

This can and does work well at multiple institutions with extensive robotics curricula that relies on ROS being installed. This formulation also works well for students in clubs where they use ROS (like robotics or motor vehicle clubs) and will be using on their systems for multiple semesters. Still, this takes significant time and effort to do for many students in class and, as mentioned before, there is a risk to damaging the computer and could be overkill for the needs of the course. Additionally, this process may not be equitable to students who have access to laptop computers with various amounts of computational power.

3.2 Linux via Virtual Machines (VM) / Container

3.2.1 Overview of Implementation

There are multiple VM options, such as VMware or Parallels that can create Linux based virtual machines. The VM set ups can be preset for students, to minimize struggle in starting to use ROS (as installation can be a major barrier to using these in classes). These VMs can be run on students' local machines or remotely if a university has virtual computational resources available for courses. This approach is used at multiple universities, some of which noted the utility of familiarizing students with using VMs, as they are very popular in industry[10]. Still, VMs, when run locally, have a limit to the processing power they can be run at. VM implementation is best used in courses where tasks require limited processing power, unlike common physics engines, like Gazebo which can run prohibitively slow[12].

3.2.2 Example Implementation: External VM

For example, ROS can be preinstalled with other packages like Gazebo, Moveit, and example robots in a ubuntu VM and then distributed to students. The can visualize the VM using FastX. The amount of RAM/storage/computational power is allocated by the university IT department and can therefore be uniform across students. An example implementation of this approach can be seen in this gitlab repository [14].

3.2.3 Example Implementation: Internal VM

In the case that an engineering department already has computers and/or IT support available for students, perhaps in a computer lab, it is straightforward to offer students access to ROS via VM's. One option that has become increasingly popular is through Parallels. A step by step git to setup Parallels on MAC with ROS2 on a machine is described in this git repository [15].

3.2.4 Recommendations

Using VMs to use ROS in the classroom has the main advantage that computers can go back to their original state (in terms of partitioned hard drive) very easily after a course is completed. VMs can also allow students to try different versions of ROS or other programs that work with different operating systems with the same access to computational power when each are running. Additionally, with VMs that are run externally by universities computational resources, students can have equivalent access to computational resources regardless

of what local computer they are using, which can be more equitable. Additionally, VMs can be pre-setup by instructors to make sure all students have the same packages; setup is more efficient for students and less trouble shooting for course staff.

Still, VMs are limited in computational power by either the local allocation of the computer (local VMs) or the allocated resources of the institution (external VMs), which are also reliant on sufficient access to wifi. Additionally, VMs can be less intuitive to connect to hardware.

3.3 ROS for Windows

Some users may prefer to run ROS on Windows machines. ROS2 is officially supported for Ubuntu, Fedora, Windows and macOS. ROS1 has experimental builds for Windows. In both cases, ROS can be built for Windows using Visual Studio and the Windows package manager chocolatey. This allows users to avoid partitioning or wiping existing hard drives that run Windows, and to run ROS in a familiar environment. Official installation guides can be found online for ROS 1 and ROS 2 [16][17].

3.4 ROS for Mac

Similarly, ROS2 can be installed on macOS with Xcode and the package manager brew. ROS for Mac is a new development and less widely used and supported. However, this may be attractive for Mac users hesitant to dual-boot or abandon macOS. As with Windows, Mac installation instructions are available in the official documentation [16][17].

3.5 RoboStack

ROS was originally developed in C++, but as Python has increased in popularity and use, so too has ROS development in Python. RoboStack was developed by Open Robotics as a bundling of ROS that can be deployed on Linux, Mac, or Windows OS via the Python package manager conda [18]. While not all ROS packages are available via RoboStack, this is a lightweight, easily deployable option, especially for those who are hesitant to switch OS or to build experimental ROS distributions. Detailed installation instructions are available in the RoboStack Github [19].

4 Choosing ROS 1 vs ROS 2 for Robotics Courses

4.1 ROS 1

The largest benefit of using ROS1 is the access to legacy code and programs that have yet to migrate to ROS2. Functionally, ROS1 and ROS2 work similarly for many course projects, but students and instructors would have access to extensive legacy code and documentation if they choose to use ROS1. Additionally, there are some hardware and programs that have not updated all their packages to work with ROS 2 yet. Even for those that have migrated, many times these packages and related functionalities have limited documentation and examples [20].

4.2 ROS 2

ROS2 core difference with ROS1 is the improved communication stack using real-time data distribution service (DDS) protocol. DDS improves real-time communication, scalability and security performance compared to ROS1 and is more resilient to network issues with message delivery in distributed systems with many robots and/or sensors. Additionally, ROS1 is slated to be unsupported by OpenRobotics in May 2025[20]. For robotics courses that share hardware with other courses, ROS2 also uses the latest version of Ubuntu, which initiated the switch for some courses [11]. Additionally, many robotics startups have already migrated to ROS2 [21], with it expected to be the robotics community standard in the near future[20].

5 Drawbacks to ROS in Undergraduate Curricula

In his analysis of robotics in post-secondary education, Esposito identifies several major drawbacks to implementing ROS [13]. Esposito compared the use of robotics software and programming languages in the classroom like ROS, MATLAB, and C. He found that ROS adoption significantly lagged the other more mature software. ROS is also substantially more complicated than software like MATLAB. Students need to learn to navigate a Linux environment, use object-oriented programming concepts, and link various languages and libraries in their development environment [13]. Robotics, being so interdisciplinary, may draw students who are not particularly strong in or interested to develop the computer programming skills required to be successful with ROS. Finally, with so many options for implementation, instructors may find it difficult to support students across the many options. Thus we recommend that any instructors interested to implement ROS in their courses should delineate very specific implementation details for students. For example, the instructor might ask that students only use ROS as a virtual machine on university computers in a designated classroom. The instructor can verify the setup ahead of time, provide detailed instructions for students,

and guarantee access to ROS regardless of the students' access to appropriate personal hardware.

MATLAB remains very popular in undergraduate robotics programs. It is well-supported across many platforms, and most instructors are familiar with the software. Even programs that offer courses with ROS tend to offer the majority of courses based on MATLAB.

6 Conclusions

ROS is steadily growing in popularity and adoption [1]. However, incorporating ROS into undergraduate curriculum is far from trivial. Despite the steep learning curve and barriers to entry around hardware and software, there are many promising solutions. Courses designers and instructors should familiarize themselves with the implementation options elucidated above. There are trade-offs between each option, and the best solution should be tailored for individual institutions and their needs. Some institutions may be able to devote dedicated Linux computers for students to use on-campus. Other institutions may be able to support VM deployment or even remote access to Linux servers. There are promising new options for installing ROS directly to student laptops, even those running Windows and Mac. Newer deployment options, like ROS2 or Roboflow may have less documentation, developer support, and reliability than older options, however these may not be barriers depending on the course scope and goals. With that said, there may be courses that are better served by other software entirely. ROS mastery requires significant time and energy apart from that required to master robotics and other STEM concepts. However if a student aspires to a career in robotics research and development, ROS mastery is likely well worth the initial investment.

References

[1] K. Scott and T. Foote, "2022 ROS Metrics Report," Available at http://download.ros.org/downloads/metrics/metrics-report-2022-07.pdf (2024/02/07).

[2] T. M. Santos, D. G. S. Favoreto, M. M. d. O. Carneiro, M. F. Pinto, A. R. Zachi, J. A. Gouvea, A. Manhães, L. F. Almeida, and G. R. Silva, "Introducing robotic operating system as a project-based learning in an undergraduate research project," in *2023 Latin American Robotics Symposium (LARS), 2023 Brazilian Symposium on Robotics (SBR), and 2023 Workshop on Robotics in Education (WRE)*, 2023, pp. 585–590.

[3] J. Kerr and K. Nickels, "Robot operating systems: Bridging the gap between human and robot," in *Proceedings of the 2012 44th Southeastern Symposium on System Theory (SSST)*, 2012, pp. 99–104.

[4] R. P. Salas and J. Ho, "A remote/virtual robotics lab," in *2021 IEEE Frontiers in Education Conference (FIE)*, 2021, pp. 1–4.

[5] E. Tosello, S. Michieletto, and E. Pagello, "Training master students to program both virtual and real autonomous robots in a teaching laboratory," in *2016 IEEE Global Engineering Education Conference (EDUCON)*, 2016, pp. 621–630.

[6] T. Tsoy, L. Sabirova, R. Lavrenov, and E. Magid, "Master program students experiences in robot operating system course," in *2018 11th International Conference on Developments in eSystems Engineering (DeSE)*, 2018, pp. 186–191.

[7] L. Joseph and J. Cacace, *Mastering ROS for Robotics Programming: Best practices and troubleshooting solutions when working with ROS*, 2021.

[8] J. Grönman, M. Saarivirta, T. Aaltonen, and T. Kerminen, "Review of artificial intelligence applications in the ros ecosystem," in *2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO)*, 2021, pp. 1149–1153.

[9] A. B. M. Pereira and G. S. Bastos, "Rosremote, using ros on cloud to access robots remotely," in *2017 18th International Conference on Advanced Robotics (ICAR)*, 2017, pp. 284–289.

[10] E. Sarda and B. Hament, "ROS in Undergraduate Curriculum at Lake Superior State University Interview," 2023.

[11] S. Oca, B. Hament, and F. Ferland, "ROS in Undergraduate Curriculum at Université de Sherbrooke Interview," 2023.

[12] S. Oca, B. Hament, J. Dawkins, M. Kutzer, E. Rodriguez-Seda, J. Esposito, and J. Piepmeier, "ROS in Undergraduate Curriculum at United States Naval Academy Interview," 2023.

[13] 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.

[14] J. Dorff, "Ansible-ros-desktop," https://gitlab.oit.duke.edu/oitde/ansible-ros-desktop, 2023.

[15] S. Cui, "MacOS VM Setup," https://github.com/Amikatoi/Compliant_motion_control/tree/main/macos_vm_setup, 2023.

[16] G. Staples, "ROS 1 Windows Installation," Available at https://wiki.ros.org/Installation (2023/12/12).

[17] G. Staples., "ROS 2 Documentation," Available at https://docs.ros.org/en/humble/Installation.html (2023/12/12).

[18] T. Fischer, W. Vollprecht, S. Traversaro, S. Yen, C. Herrero, and M. Milford, "A robostack tutorial: Using the robot operating system alongside the conda and jupyter data science ecosystems," *IEEE Robotics and Automation Magazine*, 2021.

[19] T. Fischer, “Getting Started with RoboStack,” Available at https://robostack.github.io/GettingStarted.html (2023/12/12).

[20] T. Cappellari, “The ros1 vs ros2 transition,” https://www.swri.org/industry/industrial-robotics-automation/blog/the-ros-1-vs-ros-2-transition, 2022.

[21] S. Cui, “Ros robotics companies,” https://github.com/vmayoral/ros-robotics-companies?tab=readme-ov-file, 2023.