Work in Progress: Introductory Mobile Robotics and Computer Vision Laboratories Using ROS and MATLAB

Mr. Robert L. Avanzato, Pennsylvania State University, Abington

Robert Avanzato is an associate professor of engineering at the Penn State Abington campus where he teaches courses in electrical and computer engineering, computer science, and information sciences and technology. His research interests are mobile robotics, computer vision, intelligent systems, virtual world technology and innovative education.

Mr. Cullen G Wilcox, Penn State Great Valley

Cullen Wilcox is a senior in the B.S. Engineering, Multidisciplinary Engineering Design major at Penn State Abington. His works focuses on robotic autonomy through ROS and MATLAB. Over the course of the past two years Cullen has worked with robots such as the TurtleBot and Parrot Bebop using ROS. Recently his work focused on using multiple robots. He hopes to continue this work in his career.
Introductory Mobile Robotics and Computer Vision Laboratories
Using ROS and MATLAB

Abstract

Robot Operating System (ROS) is an open source, Linux-based robotics development and deployment system which supports many commercial and research and development robots. The educational advantage of using the MATLAB interface to ROS is to provide students with a more intuitive and interactive programming environment, visualization tools, and integration of other MATLAB toolboxes such as computer vision and control. Laboratory exercises were developed and tested using MATLAB Robotics Systems Toolbox and ROS-enabled Turtlebot robots, as well as a low-cost, autonomous, indoor quadcopter. The educational goals of providing an introduction to mobile robots, sensors, computer vision and quadcopter technology were satisfied and the student reaction was positive. There are two key results. One result is that students with a wide range of programming experience are supported in this MATLAB-based development model. The second result is that multiple hardware platforms can be programmed using the same software tools. These results will be of interest to educators introducing low-cost, yet sophisticated robotics lab experiences into a diverse range of undergraduate engineering courses. These results also have significance to the introduction of robotics concepts in K-12 and STEM activities.

Introduction

Robot Operating System (ROS) is an open source, Linux-based robotics development and deployment system which supports many commercial and research and development robots, including the Turtlebot, Husky, PR2, Baxter and as well as a variety of industrial and educational robots [1]. ROS provides a distributed, networked, message-passing system that provides a standard development and deployment software architecture across a variety of sensor and hardware platforms. However, the learning curve for implementing ROS solutions is steep for undergraduate engineering students who possess diverse backgrounds and skills in software development. Part of the motivation behind ROS is to address software reuse in robot systems and to provide a consistent framework for many robotics platforms and for systems which integrate many diverse robotics subsystems, both in hardware and software. As ROS grows in popularity and usage, it is important that students be exposed to ROS in the engineering undergraduate curriculum in an efficient, low-cost and effective manner.

Although ROS was largely limited to use at the graduate level, there is a growing interest on the part of engineering faculty to integrate ROS into the undergraduate curriculum. Undergraduate research projects using ROS and Turtlebot mobile robot platforms are described in Wilkerson [2]. ROS was used in the study of a robotic arm to introduce kinematics to undergraduate students and is presented in Yousuf [3, 4]. ROS control for a robotic arm for balancing a ball on
a plate was developed for use as a teaching tool for laboratory courses in Khan [5]. Luo [6] discusses a multi-laboratory approach to teaching ROS to electrical engineering students in the undergraduate and graduate levels.

As can be observed from the literature, the general emphasis at this time is to employ ROS for special topics or research topics in the undergraduate curriculum or to limit exposure to ROS to students who possess the requisite Linux, Python and/or C++ skillset. In order to integrate ROS more fully into the curriculum, and across engineering disciplines, another approach is needed.

In 2015, MATLAB released the Robotics System Toolbox which provides a ROS interface and associated robotics algorithms and tools. This MATLAB product enables engineering students, especially in an introductory course, to more easily communicate with any ROS-enabled robots from standard Windows OS and/or Mac OS workstations running MATLAB. The advantage of this MATLAB solution is to provide students with a more intuitive and interactive programming environment, visualization tools, and easy integration of other MATLAB toolboxes such as computer vision (instead of OpenCV, for example) and control.

This paper describes a multi-year study in which the MATLAB Robotics Toolbox ROS interface was integrated into a senior-level introductory robotics design course and a computer vision course in a multidisciplinary engineering design option of a B.S engineering program at Penn State Abington. Laboratory exercises were developed and tested using MATLAB Robotics Systems Toolbox and ROS-enabled Turtlebot 2 robots from Clearpath Robotics, as well as a low-cost indoor quadcopter (Parrot Bebop 2). Topics covered in the laboratories were primarily in the areas of robot navigation, feature detection and object tracking. The primary significance of this case study is that, based on our experiences, development and deployment of algorithms for robot navigation and computer vision using ROS were accomplished with a less steep learning curve and with less coding complexity using the MATLAB-based approach than in the C++/Python/Linux environment. Secondly, the students were able to control two diverse robot platforms (a mobile robot and a quadcopter) using the same software tool and environment. Student reaction to the use of the MATLAB ROS interface was positive overall. A description of the set of the labs, sample code and best practices will be discussed and also the use of the Gazebo 3D simulation to supplement the robotics activities. Laboratories involving the control of autonomous indoor quadcopter with MATLAB will be presented as well. It is anticipated these results will be of interest to educators introducing advanced robotics labs into a diverse range of undergraduate engineering courses with students possessing a wide range of programming backgrounds (including students with limited programming experience). These results also have significance to the introduction of modern robotics concepts in K-12 and STEM activities.

2. Mobile Robot Laboratory Development

This section will describe the project-based mobile robot laboratories developed to satisfy our educational objectives of introducing mobile robot concepts and algorithms using ROS in a MATLAB environment which does not require the class of students to have expertise in Python
or C++ or Linux. The advantage of this approach is that the robotics labs (as well as the introduction for computer vision labs) can be adapted into a variety of educational environments including lower-division engineering courses and K-12.

We used the Turtlebot 2 ROS-enabled robot from Clearpath as our mobile robot platform. The Turtlebot supports differential-drive operation, odometry sensors, and features a Kinect sensor to provide a video signal as well as a depth camera to provide a lidar-type signal. The Turtlebot 2 requires a dedicated netbook running Linux 14.02 and ROS (Indigo) and is connected wirelessly (Wi-Fi) to a router. A Windows laptop running MATLAB is also connected to the router and communicates through Wi-Fi to the Turtlebot. All of the software development, data visualization and analysis is performed by the students in the MATLAB environment. At the start of the lab, ROS and the Turtlebot drivers are initiated on the Turtlebot robot as well as the network connection. In our laboratory setup, the students did not have to access the Linux netbooks on the Turtlebots in any way.

Here is a diagram (figure 1) of the Turtlebot network configuration in our laboratory setting.

![Turtlebot network configuration diagram](image)

**Figure 1: Turtlebot network configuration**

Below is a list of mobile robotics laboratories developed to serve as a module in an introduction to mobile robotics, sensors and mapping in a senior-level robotics design and applications course for B.S. Engineering students. The first set of labs (Lab #1 to #4) introduce the students to the ROS/MATLAB environment and the basic kinematics and sensors of the mobile robot platform. Labs #5 and #6 are project-based and require students to work in teams to explore more advanced robot algorithms to satisfy course objectives. All labs were completed with the physical Turtlebot and also in the Gazebo simulation software. Each lab (or set of labs) can be completed in a standard laboratory session.

- Lab #1: Move Turtlebot forward for 10 seconds
- Lab #2: Move Turtlebot forward 6 feet using odometry
• Lab #3: Move Turtlebot in a square path (and a circular path)
• Lab #4: Move Turtlebot to move forward until obstacle is detected
• Lab #5: Avoid obstacle using lidar (introduce path planning)
• Lab #6: Turtlebot mapping and localization (team projects exploring obstacle avoidance using vector field histograms and other advanced algorithms)

An example of MATLAB code to move the Turtlebot forward for 10 seconds is shown in figure 2. This is considerable less code and less steep learning curve than would be necessary in Python or C++ in a Linux environment.

```matlab
ipaddress = '192.168.1.1'    % IP of Turtlebot (will depend on your setup)
rosinit(ipaddress)     % start ROS
robot = rospublisher('/mobile_base/commands/velocity');      % publish velocity topic
velmsg = rosmessage(robot);     % get message format for velocity
tic;                          % tic initializes the timer
while (toc < 10)               % loop until toc = 10 seconds
    velocity = 0.5;        % meters/second
    velmsg.Linear.X = velocity;
    send(robot,velmsg);
end % end while loop
velocity = 0;                % set velocity to 0 to stop robot
velmsg.Linear.X = velocity;
```

Figure 2: MATLAB code to move ROS Turtlebot forward

Robot simulation in Gazebo [7] software was also required for each lab. Gazebo is a 3D simulation which supports the Turtlebot and other ROS platforms. Gazebo was installed in a Linux shell (VMware) on the same Windows computers as MATLAB. The same MATLAB ROS code that operated the simulated robot in Gazebo can be used to operate the real Turtlebot. The only difference is the change in one line of the MATLAB code to connect to the IP address of the Gazebo environment versus the IP address of the real robot. Figure 3 depicts the Gazebo environment with a Turtlebot in a virtual room with obstacles.
An introduction to computer vision was also delivered to the students which covered topics in basic image processing techniques such as dilation, erosion, open, close, blob analysis, linear filtering, edge detection, Hough transforms and object detection. In this introductory computer vision module, color detection and color tracking experiments were performed with the ROS robots and MATLAB. Because the students had familiarity with the Turtlebot robot and ROS, it was efficient to include computer vision activities using the Turtlebot when covering computer vision topics. More importantly, the students were developing robot algorithms applying ROS commands and using functions in the Computer Vision Toolbox in the same MATLAB development environment.

3. Quadcopter Laboratory Development

One of the advantages of ROS is that it provides a common development framework across a variety of sensors and hardware platforms. In this section, we will describe ongoing laboratory experiences using MATLAB ROS to command the Parrot Bebop 2 quadcopter (drone). This quadcopter is low cost (~ $300 USD) and is capable of flying indoors and outdoors with a 20-minute battery life per charge (see figure 4). Because of its small size, and the availability of ROS support, this drone is an ideal platform for use indoors in an educational laboratory settings. (It should be noted that all of our experiences were restricted to indoor use.) The educational objective of these laboratory experiences is to expose students to the programmatic control of quadcopter technology using MATLAB and also to introduce computer vision as applied to autonomous quadcopter navigation. The ROS package for the Parrot Bebop (bebop_autonomy) was developed by a 3rd party [8] and we were first introduced to the Bebop ROS technology in Fairchild [9] which contains a variety of very useful ROS robotics projects in a Python and Linux software environment.

The introductory student labs developed for the quadcopter module include: 1) take-off and landing under MATLAB control, 2) moving the quadcopter forward a fixed distance using time and odometry, 3) moving the quadcopter in a fixed pattern, such as a square, 4) using
computer vision to track a colored disk and 5) experimentation with autonomous landing using colored markers on the floor and also markers positioned on mobile robot platforms (figure 5). A key result here is that the students used equivalent ROS commands in MATLAB to control both the Turtlebot and the Parrot Bebop quadcopter, thus allowing for labs to be delivered in a highly efficient and effective manner. (We did observe that the turning (pivoting) of the quadcopter using the odometry function was imprecise in our experiments.)

We have successfully demonstrated two architectures for operating the Parrot Bebop quadcopters with MATLAB and ROS running in a Windows environment. In one architecture, a Linux shell with ROS was installed as a virtual machine on the Windows host laptop. The host machine also connected to the Wi-Fi of the Bebop. This allowed for full communication and control of the Bebop and the Turtlebot in addition to receiving a video stream from the quadcopter to be analyzed in MATLAB. In our second architecture, we connected both the MATLAB host laptop (without the Linux virtual machine) and a Turtlebot robot to the Bebop and we were able to demonstrate MATLAB control over both platforms simultaneously through one MATLAB session on the host student computer (figure 6). Coordinated control of both platforms was possible but the video stream to the MATLAB laptop from the quadcopter had a high latency (2 to 3 seconds) limiting its use for computer vision experiments. However, in the first architecture with the virtual Linux machine, the video feed from the quadcopter to MATLAB was sufficiently fast (latency < 0.5 seconds) to do basic experiments with color tracking and quadcopter landing on colored markers autonomously. The performance was demonstrated to be more than satisfactory for educational purposes.

Figure 4. Parrot Bebop 2 Quadcopter (drone)
Figure 5: Quadcopter detection of red object positioned on Turtlebot

Figure 6: Bebop quadcopter video with MATLAB laptop and Turtlebot netbook
Below is sample MATLAB code to command the Bebop quadcopter to take-off, hover for 5 seconds, then land. As can be seen, MATLAB offers a simple and intuitive interface to ROS features.

```matlab
[cmd,msg] = rospublisher('/bebop/takeoff', 'std_msgs/Empty');
send(cmd,msg);
pause(5)
[cmd,msg] = rospublisher('/bebop/land', 'std_msgs/Empty');
send(cmd,msg);
```

4. Conclusions and Future Directions

This paper outlined a strategy to introduce students to mobile robotics and computer vision using ROS and MATLAB tools in a lab setting. The objectives were to develop hands-on laboratories using ROS-enabled robot platforms, 3D simulation tools and MATLAB’s ROS interface to introduce robotics concepts and algorithms. The overall result combines the advantages and scalability of ROS with the ease-of-use and power of MATLAB programming environment with the toolbox support. Students with a wide range of prior programming experience were able to fully participate in the labs and focus on the robot technology and algorithm development instead of complex software features, drivers, connectivity issues, and operating system issues. Using ROS does not hide any required concepts or applied theory covered in the course. On the contrary, the use of ROS and MATLAB allows students to focus more on the robot algorithms, sensor data collection and visualization, which serves the robot course objective of emphasizing applications and robot system design. ROS tools are equivalent across ROS-enabled robot platforms, so students can program one robot platform using the same software interface as is used to program a different hardware platform. This common set of interface tools also exists for a variety of sensors across various platforms. This concept of a common software interface was demonstrated by the successful programming of the Turtlebot mobile robot platform and the Parrot Bebop quadcopter platform. Additionally, this capability enabled students to experiment with multi-robot coordination, which also supports the course objective to expose students to diverse robot platforms and systems. This overall approach allows for efficient, low-cost, effective and scalable delivery of robotics education laboratories. There is, of course, some initial overhead in installing the ROS software and drivers and the virtual machines and establishing proper connectivity to the robots. We plan to assess improvements in student learning in future studies. However, we do note that the use of the MATLAB/ROS tools has allowed the students to participate in more advanced robotics labs and projects than was the case before using these tools. More specifically, there was an opportunity to include multiple robot platforms (Turtlebot and quadcopters) into a single-semester lab experience which was not practically possible in the past without the use of these tools.

The robotics lab experiences included in this paper were at an introductory level and could be adapted and delivered to students in lower division courses, or as an introductory experience in an upper division course followed by more advanced projects using the same software and
hardware tools. These robotics labs using the Turtlebots have been tested with small group (6-10 students per class) of senior-level students in our B.S. Engineering (Multidisciplinary Engineering Design option) at Penn State Great Valley location over the past 2 years and the student feedback has been positive. The student testing with the quadcopter labs in a class setting has started in spring 2018, and we plan to collect additional data. We believe the results will scale well to use in larger classes due to the easily accessible programming interface (MATLAB) and the low-cost of the equipment. We also plan to develop more ROS labs using MATLAB in the area of multi-robot control and also for low-cost robot manipulators. It is hoped that these experiences and strategies will be useful for educators, both undergraduate and K-12, who wish to advance robotics education for students.

5. References


