Project Based Learning Using the Robotic Operating System (ROS) for Undergraduate Research Applications

Dr. Stephen Andrew Wilkerson P.E., York College PA

Stephen Wilkerson (swilkerson@ycp.edu) received his PhD from Johns Hopkins University in 1990 in Mechanical Engineering. His Thesis and initial work was on underwater explosion bubble dynamics and ship and submarine whipping. After graduation he took a position with the US Army where he has been ever since. For the first decade with the Army he worked on notable programs to include the M829A1 and A2 that were first of a kind composite saboted munition. His travels have taken him to Los Alamos where he worked on modeling the transient dynamic attributes of Kinetic Energy munitions during initial launch. Afterwards he was selected for the exchange scientist program and spent a summer working for DASA Aerospace in Wedel, Germany 1993. His initial research also made a major contribution to the M1A1 barrel reshape initiative that began in 1995. Shortly afterwards he was selected for a 1 year appointment to the United States Military Academy West Point where he taught Mathematics. Following these accomplishments he worked on the SADARM fire and forget projectile that was finally used in the second gulf war. Since that time, circa 2002, his studies have focused on unmanned systems both air and ground. His team deployed a bomb finding robot named the LynchBot to Iraq late in 2004 and then again in 2006 deployed about a dozen more improved LynchBots to Iraq. His team also assisted in the deployment of 84 TACMAV systems in 2005. Around that time he volunteered as a science advisor and worked at the Rapid Equipping Force during the summer of 2005 where he was exposed to a number of unmanned systems technologies. His initial group composed of about 6 S&T grew to nearly 30 between 2003 and 2010 as he transitioned from a Branch head to an acting Division Chief. In 2010-2012 he again was selected to teach Mathematics at the United States Military Academy West Point. Upon returning to ARL's Vehicle Technology Directorate from West Point he has continued his research on unmanned systems under ARL's Campaign for Maneuver as the Associate Director of Special Programs. Throughout his career he has continued to teach at a variety of colleges and universities. For the last 4 years he has been a part time instructor and collaborator with researchers at the University of Maryland Baltimore County (http://me.umbc.edu/directory/). He is currently an Assistant Professor at York College PA.

Dr. Jason Forsyth, York College of Pennsylvania

Jason Forsyth is an Assistant Professor of Electrical and Computer Engineering at York College of Pennsylvania. He received his PhD from Virginia Tech in May 2015. His major research interests are in wearable and pervasive computing. His work focuses on developing novel prototype tools and techniques for interdisciplinary teams.

Lt. Col. Christopher Michael Korpela, United States Military Academy

LTC Christopher Korpela is an Academy Professor serving as the Deputy Director of the Electrical Engineering Program. His previous military assignments include: Tank Platoon Leader, Scout Platoon Leader, Troop Executive Officer, Squadron Adjutant, and Squadron Assistant Operations Officer in 1st Squadron, 3rd Armored Cavalry Regiment. During a brief break in service, he worked in the civilian sector as a hardware engineer for National Semiconductor Corporation. He deployed as the Headquarters Commander for the 439th Engineer Battalion (USAR) while attached to 2nd Brigade, 82nd Airborne Division in Baghdad, Iraq, in support of Operation Iraqi Freedom. In 2010, he served as the 2nd Infantry Division Network Engineer at Camp Red Cloud, South Korea. During the Summer of 2015, he deployed with the 82nd Airborne Division in support of Operation Inherent Resolve. LTC Korpela is a graduate of the Armor Officer Basic Course, Engineer Captains Career Course, Combined Arms and Services Staff School, Command and General Staff College, Ranger School, Airborne School, and Air Assault School. His research interests include robotics, aerial manipulation, and embedded systems.

Project-Based Learning Using the Robotic Operating System (ROS) for Undergraduate Research Applications

Project-based learning (PBL) has been shown to be one of the more effective methods teachers use in engineering and computer science education. PBL increases the student's motivation in various topic areas while improving student self-learning abilities. Typically, PBL has been employed most effectively with junior- and senior-level bachelor of science (B.S.) engineering and computer science students. Some of the more effective PBL techniques employed by colleges and universities include robotics, unmanned air vehicles (drones), and computer science-based technologies for modeling and simulation (M&S). More recently, an open-source software framework for robotic and drone development, called the Robot Operating System (ROS), has been made available through the Open Source Robotics Foundation. While not an actual Operating System (OS), ROS provides the software framework for robot software and associated hardware implementation. In this paper, we examine the use of ROS as a catalyst for PBL and student activities in undergraduate research. ROS provides students, after some time investment, with the ability to develop robotic capabilities at a high level. Moreover, ROS allows a building-block approach to robotics research. The results and "how-to" data from our projects are provided on GitHub to accelerate future efforts with other PBL learning endeavors. A results-based evaluation criteria will be used as a partial measure of merit. To this end, we post usage data from cited repositories as evidence of the contribution. We will also contrast expenditure of time and effort vs. a traditional classwork environment while coupling some measure of comprehension and mastery of the underlying research topics used by the students in their undergraduate research topic.

Introduction

Robotics has, for the past several decades, been a mainstream staple for project-based learning (PBL). PBL and robotics have been used at every level of education to spark student imagination and learning activities. With First Robotics (Moylan 2008, Barron et. al. 2008) and VEX (Das Shuvra et.al. 2010, Ruzzenente 2012) at levels from K1–12 (Grandgenett 2012) to undergraduate programs, and (Sébastien 2001) we have seen the value of PBL when used with Robotics. In these examples, many of the project learning activities are centered around competitive goals. These goals, however, are not a necessity of PBL. The same learning objectives can also be achieved by incorporating them into a specific design objective or project requirements. Using PBL and robotics, (Ramos and Espinosa) showed that the same learning objectives of a traditional classroom could be achieved via PBL. The crux of their approach was to build a student-centered learning model using PBL that served as a bridge from a teacher-led environment to one of self-discovery. (Hees et al. 2009) also showed the value of self-discovery and learning with minimal input from instructors. Regardless of the classroom objective, robotics seems to be a topic in which students will devote time and energy learning new materials to accomplish specific tasks or goals.

Robotics is a multi-discplinary field incorporating elements of mechanical and computer engineering, and computer science. Traditionally, robotics courses and degrees have typically been offered through graduate programs but has seen an expansion into the undergraduate curriculum through capstone projects (Michalson 2010), upper-class courses (Keer 2012, Meuth 2009, Garcia 2015, Lessard 1999) and freshman engineering (Xu et. al 2014) through introductory platforms such as the Lego NXT and Vex robotics. Given the increased number of incoming students interest in robotics, two undergraduate programs have been developed in robotics engineering (WPI 2017, Lawrence Tech 2017). These programs have observed a large growth in enrollment and successful placement of students in industry or graduate school (Gennert 2013). Within these independent studies we focus on the use of the Robotic Operating System (ROS) to facilitate robot design and implementation. ROS is free, open-source and supplies a large ecosystem of nearly 3,000 packages to accelerate application development. ROS has also been used in education settings to teach kinematics to mechanical engineering students (Yoursuf 2015) and develop robotic arms with high school students (Yousef 2016).

In this work, we used some basic proven principles while attempting to raise the bar of difficulty and examine if undergraduate students might use robotics, in particular ROS, to learn advanced concepts while contributing to undergraduate research activities. While we fully understand that this question will not be laid to rest by this study into the topic, we are examining on a limited basis where this should work. Our initial fear was in overreaching and thereby providing student frustration rather than meaningful learning experience. However, once we started the process, we found these potential problems not to be an issue for the students selected for this study.

To begin, it is appropriate to give some of the background for ROS so that others may determine whether this is an appropriate platform for their programs. ROS has become a huge project in the past 5–7 years, and it has many contributors. Originally, several efforts at Stanford University that involved artificial intelligence were developed into an in-house software system that could be used for robotics. Later, a small start-up company called Willow Garage provided essential resources to extend the initial concepts developed at Stanford. The project was furthered by numerous researchers from around the world who contributed software and hardware examples to the core ROS concept and basic functionality. Today, ROS is used at numerous universities, government labs, and elsewhere around the world for robotics research. While ROS is a staple of most graduate robotics programs, it is only now starting to be used in undergraduate programs. Additionally, ROS is widely used for computer science programs and exposes students to best practice with a number of computer programming paradigms. In this study, we take advantage of these features while using the basic ROS framework to expose students to hardware and software integration techniques that are usually reserved for graduate programs. Furthermore, we use ROS with PBL to expose students to practical problems found in robotics while expanding their knowledge in control methods, vision algorithms, and electronic integration of components needed for our project.

Our overall goal of this study was to expose students to advanced topics in robotics using PBL and to see if they could make meaningful progress. Our assessment criteria were not intended to be absolute, rather selectively based on a narrow perspective from which we could grow the program while making adjustments. Participation in undergraduate research experiences has been shown to increase a student's confidence in the discipline and increase continuation into graduate school. (Conrad 2015, Russell 2007, Zydenny 2002).

Therefore, we did not open the independent study course up to simply any student wanting to work on robotic systems but instead limited the availability to only a selective few students. We targeted the students we knew had the best academic record and who had a history of going "above and beyond" on courses to get good grades. We used a short write-up of proposed topics and objectives that would be tackled during the semester. Interviews with perspective students were conducted either in person or via Skype.[ǂ] The structure of the independent study course is provided as an attachment in Appendix A. Initially, we determined to limit the first semester to three students with two professors assisting them. (In our second semester, which is starting as of the writing of this paper, we will double the number of students and add one additional faculty member.) Initially, we had one female and two male students in our course, and we met once a week formally and usually once a week informally throughout the semester. Independent study (as currently construed) addresses three of the seven "Big Ideas in Robotics" (Touretzky 2012).

Approach

For many of the topics and subtopics in this course, the students we selected had no prior knowledge or experience. In some of the topics, such as programming language, the students had already taken a course or two, introducing them to languages such as C++ and Python and giving them the basics of building, compiling, and debugging code. For larger topics, such as ROS and the Turtlebot, the students largely had little or no knowledge/awareness of the topic. However, we expect this situation to change in time as the course grows at the college and other students become aware of what we are doing through word of mouth and student social interactions. For this first semester, our probe into these topics was largely uncorrupted. Table 1 shows a survey of the students' knowledge in a number of topics that would be needed to complete the projects outlined in Appendix A as a baseline and possible partial measure of merit. Note that a "1" on the scale represents a student having no prior knowledge and a "10" represents a student being fully educated in the topic's uses and interworkings. Once again, we understand this is a subjective measure determined by a number of factors, including the students' personalities.

Each student's course of study was initially similar, but as the semester progressed, each student became more focused on his/her individual project objectives. Certainly, one of the objectives of any independent study course is to have less structured classroom time and a more open student learning initiative-based approach. After all, there are ample videos and learning materials on the web that the students could easily access to learn about the Turtlebot and ROS. Nonetheless, it was necessary to build an initial framework of knowledge from which the students could launch into individual projects while still retaining some common framework. Throughout the semester, students were encouraged to share what they had learned and openly interact with one another. To achieve a common framework of knowledge, students spent their first 6–8 weeks (basically half the semester) on a self-paced but common path. To start, the students needed to build an Ubuntu system with ROS and the Turtlebot libraries on it for the

[ǂ] Skype is a web-based application that provides video chat and other services on a variety of platforms, including Microsoft and Macintosh.

robots. To this end, we used the ASUS EeePCs 32-bit and 64-bit system architectures. The students also needed to build an Ubuntu system on their own laptops to interact with the Turtlebots. These system builds varied from dual boot to virtual systems, and each had its own set of nuances and challenges for the students.

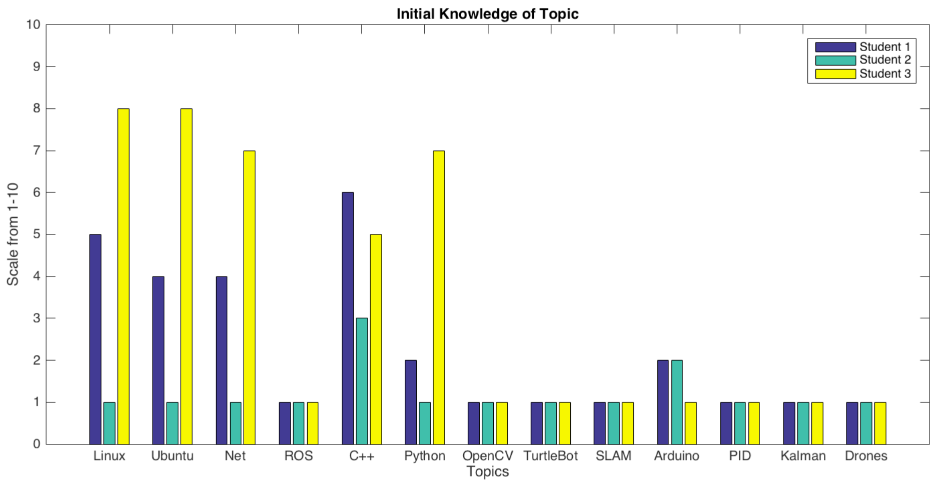

Figure 1. Initial Student Topic Knowledge.

Another objective of the initial phase of the independent study was to familiarize the students with ROS's publisher subscriber methodology and the capabilities and limitations of the Turtlebot. Students needed to rapidly understand this environment to write programs to extend capabilities already documented and available on the web. Our end state goal was to create a Turtlebot capability[†] not already available using examples that could be downloaded from the web and extended. This goal was accomplished by combining existing techniques and creating entirely new methods within the ROS environment. The Turtlebot system has more than 30 ready-made tutorials teaching students how to set up, test, and challenge their abilities. The body of these examples alone could occupy a student for the semester. However, to get started, students were required to do roughly the first dozen of these learning exercises. In a similar fashion, ROS has a number of tutorials[††] to take students from beginners to more advanced levels of programming. These examples teach students how to program in the publish subscriber environment of ROS. Once again, students were required to work the first dozen or so of these examples. Due to the level of difficulty of mastering the ROS programming environment, this process was supplemented using O'Kane's book *A Gentle Introduction to ROS*. (O'Kane 2013) Students were required to work through the first seven chapters of the book while doing the Turtlebot and ROS tutorials. The book "Practical Python and OpenCV" was also used too introduce students to the OpenCV (Rosebrock 2016) libraries for visual operations. Additionally, students were asked to give weekly oral presentations of their progress and problems, enabling open discussion with the faculty and other course students to resolve problems and present potential solutions. While this approach may seem relatively ambitious, the students were not pushed but were allowed to work at their own pace within the constraints of their own individual

† Turtlebot tutorials can be found at http://learn.turtlebot.com/.
†† ROS learning tutorials can be found at http://wiki.ros.org/ROS/Tutorials.

course loads. And what was observed was a similar willingness to spend time and energy working with the robots as what has been observed at the college on capstone projects such as Baja and Formula competitions.

Project Development

Initial growth activities included having the students all worked toward mapping the college's engineering building. The building consists of faculty offices, classrooms, labs, workshops, and study areas, which are all interconnected by a series of hallways. What was discovered in the process were numerous opportunities to expand and contribute to the growing body of examples readily available on the internet using ROS and the Turtlebot. The facility was far too large to map in one Turtlebot mission, and therefore it was necessary to either splice maps together or create a map that could be used by the Turtlebot during its missions. Other activities were broken into subprojects, and students were encouraged to work on these whenever they became stuck so that progress in their learning could continue to be made. These subprojects included ArUco tag tracking, Radio Frequency IDentification (RFID), map splicing and creation, Arduino interfacing, color tracking, and object map location determinations. Each activity was reviewed on a weekly basis to assure progress. The details of these technologies are interesting enough in themselves and are detailed in a separate publication by the students. In the end, it was proposed that the students develop a series of programs and capabilities that enabled them to have one Turtlebot navigate the building from one point to another and have another Turtlebot follow the first using Aruco tags while leaving bread crumbs. The bread crumbs were to be objects that could be found by the last Turtlebot, with each having RFID tags that identified them. The students were to work together as a team, dividing up the tasks into subtasks that each could work on. Each project was to be capable of being published as a stand-alone tutorial that could be used by other students in the future. As an added measure of merit, these projects will be documented and put on GitHub, allowing others students to "*git-clone*"[‡] them rather than starting from scratch. Furthermore, by tracking usage of the repository, the number of times other people take advantage of the work for their own projects can be ascertained. We will also use this metric as a measure of merit. The overall project was broken into three specific tasks. The level of technical difficulty also provides some measure of merit, and therefore the following text briefly provides an overview of what each task entailed. The work's technical details can be found in the GitHub repository or in the student paper.

Task 1: Mapping and Navigating.

All of the students used the Kobuki Turtlebot with an Asus laptop and either the PrimeSense[‡‡] or Xtion sensor. Both sensors use the imaging technologies developed by PrimeSense. The original Turtlebot used the Kinect sensors and also used the same technology. The imaging technology provides a three-dimensional (3D) view of the world and, along with wheel turns, helps the Kobuki Turtlebots to localize. In the Turtlebot tutorials referenced previously, there is an experiment where an operator can create a two-dimensional (2D) map and then, using that map, provide an autonomous mission for the bot. To do this, the driver would set

‡ Git-clone allows users to clone a repository into a new directory to track changes and get a head start on research needing specific capabilities. Details can be found at: https://git-scm.com/docs/git-clone.

‡‡ PrimeSense is an Israeli 3D sensing company based in Tel Aviv.

up the Kobuki to accept commands from a standard joystick on a remote computer. The computer on the bot and the local laptop are networked in the ROS environment. Then, using the data from the onboard PrimeSense camera, the bot collects vision-based data as the bot is driven around. Typically, the bot is out of sight of the operator. Then using SLAM algorithms already developed, the bot creates a map based on wheel turns and the image data being collected by the PrimeSense sensor. The data are then stored in a yaml[†‡] and pgm raster image file. A portion of a typical pgm map taken in the engineering building is provided in Figure 2. Thick black lines represent walls, white space represents free space, and gray is unknown. Each pixel on the map represents 5 cm of space. This image took about 10 minutes to create and represents about 10 m of hallway.

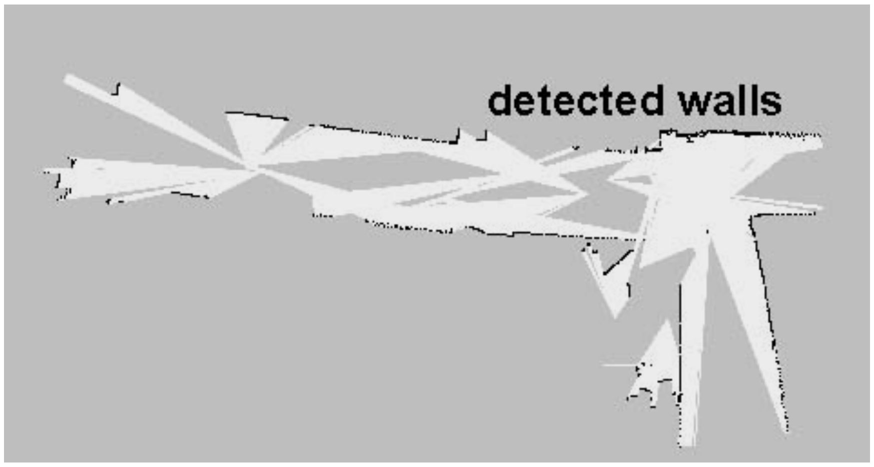

Figure 2. pgm Image File From Robot Experiments.

When the bot uses the map, if an object is put in the robot's path, it will still avoid it. When operating in the immediate area from its start point, the bot can move from one location to another without too much difficulty. However, when the maps become larger, the lines become more skewed and the bot can experience difficulty moving from one location to another. Modifying the underlying algorithms was well beyond the scope of an undergraduate PBL. Also, mapping larger areas required the maps to be spliced from separate experiments, resulting in greater uncertainty and a lower probability that the bot would actually reach its final destination. Numerous experiments were conducted with the bots. In the end, one student determined that a map of the building was needed to enable the robot to navigate longer distances. Because no program could be found on the Internet to accomplish this task, a program was needed. To do this, the student needed to develop a program that would take a raster image in any format, scale it to the appropriate dimensions, and write the associated yaml and pgm files for the Turtlebot's use. Using the tool that the student developed, others can take rough floor plans or draw their own plans to be used with the Kobuki bots and autonomous navigation missions. One of the scanned floor plan maps of the engineering building is shown in Figure 3 for reference.

The shaded area in Figure 3 represents the area from which Figure 2 was taken. The student developed the program using Python and OpenCV. Because the student had no prior experience using Python, she needed to go through a similar process to what was done with ROS

†‡ yaml stands for "yet another markup language."

and the Turtlebot. She mastered what she needed from OpenCV[‡†] tutorials and a book. The technical details, resulting code, and examples are provided via YouTube videos,[†‡‡] and the code[‡‡‡] can be downloaded from Google Drives. At the time of this writing, the code video had already been accessed 62 times.

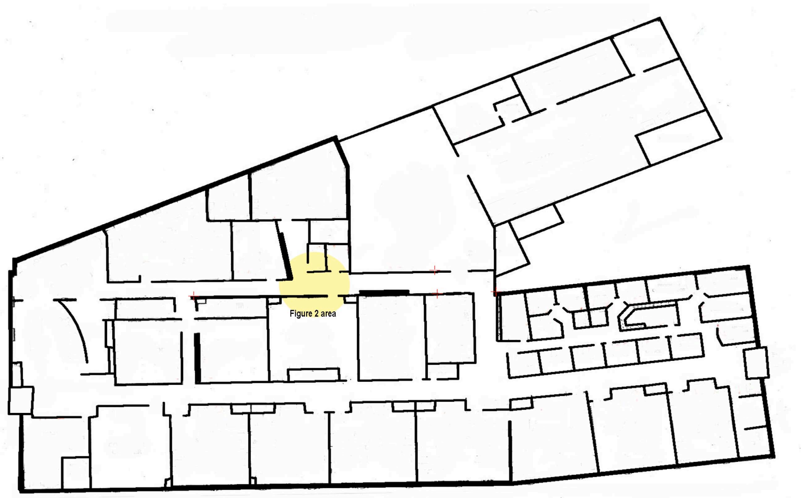

Figure 3. Engineering Building.

Task 2: Leader Follower Using ArUco Markers.

Numerous examples that use ArUco tags, their generation,[†††] and detection using OpenCV libraries[††‡] can be found on the web. However, our starting point for this project was using the GitHub example given on the USMA GitHub ROS repository. Figure 4 shows an ArUco marker being tracked and the resulting vectors describing its orientation and distance from the camera.

The vectors seen in the screen are provided as part of the ROS publisher subscriber and are accessible in the ROS publisher subscriber programming environment. What remained to be done was to write a program accessing this information and couple it with the Kobuki's movement. The program needed to use a camera to sense when the marker was moving toward or away from the bot. This need was met using the vectors shown in the figure and some control laws. For this application, the student could have used the PrimeSense camera, another universal serial bus (USB) camera, or the camera that is associated with the Asus laptop controlling the Kobuki. The student chose to use the latter but in the end added an independent camera. Additionally, this application is a control problem that requires a controller and some initial data manipulation. The issue for the data required a Kalman filter, (Haykin 2001) while the control issue could be handled using a standard proportional integral derivative (PID) controller. (Hogg et. al. 2002). In both cases, the student needed to research and experiment with these to get the

‡† http://docs.opencv.org/2.4/doc/tutorials/tutorials.html.
†‡‡ Video instructions are given at https://www.youtube.com/channel/UCQgDH1KtuoDZkafNM0QiC-A.
‡‡‡ Code can be downloaded from https://drive.google.com/file/d/0B2AcDRX3bKLVdjhPU1B2UUNRaDA/view.
††† ArUco tags can be generated via http://terpconnect.umd.edu/~jwelsh12/enes100/markergen.html.
††‡ ArUco tag detection in OpenCV is shown at http://docs.opencv.org/trunk/d9/d6d/tutorial_table_of_content_aruco.html.

robot to follow another robot smoothly. There was also the geometric issue of the leading robot turning and the following robot losing visual contact with the ArUco tag. The technical details of the effort are in the GitHub repository (at https://github.com/plynn17/Aruco_move_ros_pkg) and in the student paper. Figure 5 shows the leader follower operation in action. For this project, a C++ program (cleverly named ArUco_move) was developed in the ROS environment. The PID variables were arrived at using trial-and-error methods. In the end, the system worked reasonably well and will serve future projects, which will include the use of drone vehicles. At the writing of this paper the overview video had been viewed 47 times.

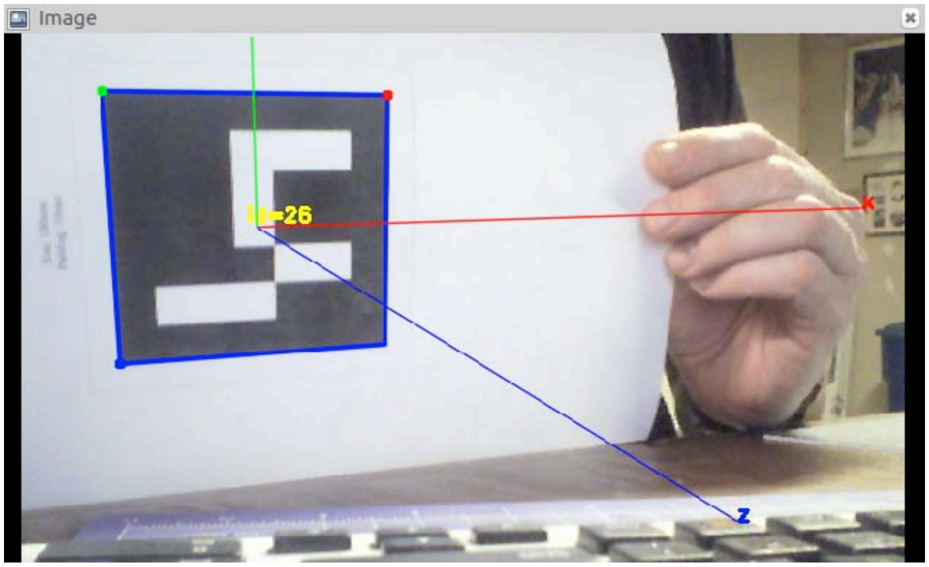

Figure 4. ArUco Marker Detection (Marker 26) using OpenCV and ROS.

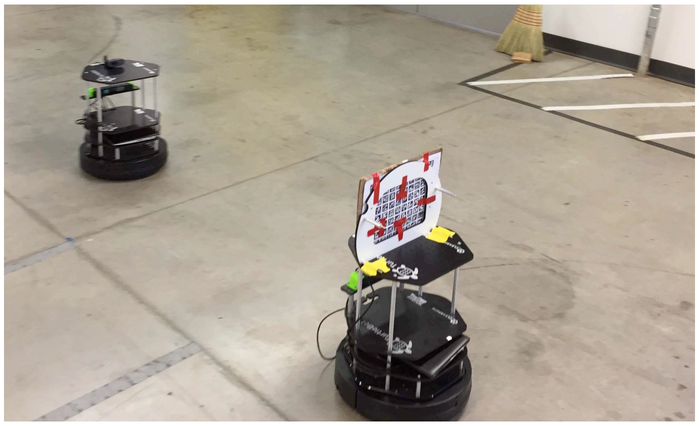

Figure 5. Leader Follower Operations.

Task 3: Object Recognition and RF Tag Detection.

Object recognition for this task was handled using Python, OpenCV, and ROS. For this task, the student needed to couple object recognition with robot movement and used the same Python and OpenCV libraries and techniques used in the first project. The only difference was that this action was conducted with a video stream rather than a single picture. However, in the end, the two methods are the same inasmuch as the video stream is treated as a series of still pictures that are being manipulated in real time. For this project, the student already had some experience writing Python scripts. However, the scripts still needed to be coupled into the publisher subscriber environment in ROS. Then the robot's movement needed to be driven by the location of the objects. The goal is to have the robot approach the object and then, using an arm, read the RF tag. The RF tag can only be read at close range; therefore, the robot movement needed to be fairly concise. For the object location, the student chose to use the PrimeSense camera rather than a USB camera. The RFID integration was far more straightforward as there are numerous how-to examples in Python on the web. The associated Python code and ROS make files for this effort can be found at https://github.com/mrjones2014/ROS-RFID-Finder.

Observations and Adjustments

(Prince 2004) points out that, when asked if Active Learning works, learning outcomes are often not available, making assessment difficult. Furthermore, when data are available, determining whether a particular approach works becomes a matter of interpretation. Our initial assessment was based on exposing students to topics that they had no prior knowledge of and seeing if they could make notable progress using PBL within an independent study course. As measures of merit, we included an initial survey of prior knowledge of the topics they would need to learn and then an exit survey, asking the same questions (see Figure 6).

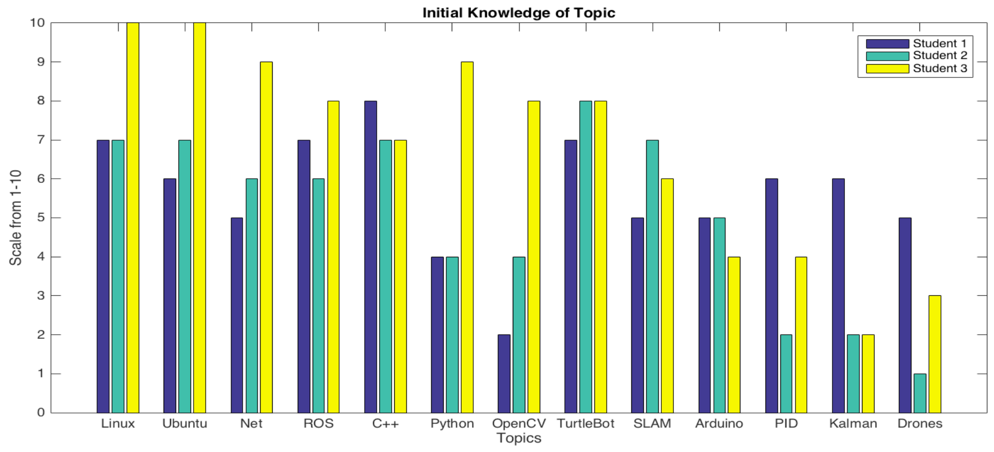

Figure 6. Final Student Topic Knowledge.

As Prince[28] points out, the inclusion of this data is subjective at best. Nonetheless, it seems prudent to have some measure, from the student perspective, of what was accomplished. In the coming semester, we will additionally track the students' time spent on a weekly basis and compare that to another one of their technical courses for comparison. We recognize that this comparison is not absolute, and students must gauge how they spend their time. Nonetheless, putting additional time into a topic or course of study will ultimately benefit them and we observed a willingness by all of our students to put additional time into this project. What was observed by the faculty was that the students were spending a large amount of their time in the lab working with the robots. Regardless of one's perspective, it can be argued that this is usually a good sign. However, in this study, we also felt that another measure of merit was whether what the students had accomplished was of interest or had value for others. For the question of "Are we making a contribution to a body of work?" a good measure is usually whether anyone else wants to use it. This measure we felt was best assessed by tracking video hits and downloads. After all, what better measure do we have then whether the community will use what we have created? Questions as to whether someone can find our examples is also of concern, but we feel that with proper tags, this concern can be mitigated. Naturally this is an ongoing process and it can be checked by visiting the links listed in this text. As one small example the map making educational videos had a combined 96 views 3 months at the time of this submission. Whether or not this is a useful measure of merit will be more evident in a year or two and published in subsequent publications.

Conclusion

For the independent study described herein, PBL was employed with some success. As mentioned, for this initial study, we were highly selective of which students we took, which we recognize likely skewed our results toward success. In the coming semester, however, we will double the number of students we allow into the course. The end goal of the course was to motivate students using robotics and drones and to see if this approach provided the same level of interest as the Baja and Formula competitions. It is well known that robotics has worked well with K–12 students in First[‡††] and VEX competitions. For our limited experiment here, we had one student apply to graduate school in a course of study of robotics. In another instance, one of our students was offered a job on the spot from a defense contractor once she described what she had been working on during the semester.

ROS is a staple of many of the graduate research programs at a variety of universities. In this paper, we have examined the use of ROS as a catalyst for PBL and student activities in undergraduate research. ROS enabled our students, after some time investment, the ability to develop robotic capabilities that would otherwise have been impossible. ROS provided a building-block template for the students who created robot behaviors and missions of interest. The results and how-to data from our projects were provided on GitHub, YouTube, and Google Drives. It is hoped that these efforts will accelerate future efforts with other PBL learning endeavors at the college and elsewhere. Results-based evaluation criteria were used as a partial measure of merit. To this end, we included citation and usage data from the YouTube videos and GitHub repository as evidence of the contribution. We also noticed an increased expenditure of time and effort vs. what is normally observed in a traditional classwork environment. Finally, as a measure of comprehension and mastery of the underlying research topics used by the students, we used before and after surveys of robotic topic areas.

References

1. Moylan, William Alexander. "Learning by Project: Developing Essential 21st Century Skills Using Student Team Projects." *International Journal of Learning,* 15.9, 2008.
2. Barron, Brigid, and Linda Darling-Hammond. "Teaching for Meaningful Learning: A Review of Research on Inquiry-Based and Cooperative Learning. Book Excerpt." George Lucas Educational Foundation, 2008.
3. Das, Shuvra, Sandra A. Yost, and Mohan Krishnan. "A 10-year mechatronics curriculum

‡†† First Robotics has multiple programs for K–12 students (see http://www.firstinspires.org/robotics/frc).

development initiative: Relevance, content, and results—Part I." *IEEE Transactions on Education,* 53.2, pp. 194–201, 2010.

4. Ruzzenente, Marco, et al. "A review of robotics kits for tertiary education." *Proceedings of the International Workshop Teaching Robotics Teaching with Robotics: Integrating Robotics in School Curriculum*, 2012.
5. Grandgenett, Neal, et al. "Robotics and Problem-Based Learning in STEM Formal Educational Environments." *Robots in K-12 Education: A New Technology for Learning: A New Technology for Learning,* 94, 2012.
6. George, Sébastien, and Pascal Leroux. "Project-based learning as a basis for a CSCL environment: An example in educational robotics." *First European Conference on Computer-Supported Collaborative Learning (Euro-CSCL 2001)*, 2001.
7. Ramos, Fernando, and Enrique Espinosa. "A self-learning environment based on the PBL approach: An application to the learning process in the field of robotics and manufacturing systems." *International Journal of Engineering Education,* 19.5, pp. 754–758, 2003.
8. Hees, Frank, et al. "Developing a PBL-based rescue robotics course." *Proceedings of the First Kuwait Conference on e-Services and e-Systems*, ACM, 2009.
9. Michalson, W., & Looft, F. (2010, June), *Designing Robotic Systems: Preparation For An Interdisciplinary Capstone Experience* Paper presented at 2010 Annual Conference & Exposition, Louisville, Kentucky.
10. Kerr, John, and Kevin Nickels. "Robot operating systems: Bridging the gap between human and robot." *System Theory (SSST), 2012 44th Southeastern Symposium on*. IEEE, 2012.
11. Meuth, R., & Robinette, P., & Wunsch, D. (2009, June), *Introducing Robots* Paper presented at 2009 Annual Conference & Exposition, Austin, Texas.
12. Garcia, J. M., & Homkes, R., & Carnes, M. T., & Taylor, K. D. (2015, June), *Lessons Learned from Team-Teaching a PBL Robotics Course with Multi-disciplinary Instructors and Students* Paper presented at 2015 ASEE Annual Conference & Exposition, Seattle, Washington.
13. Lessard, R. A. (1999, June), *Embedded Systems Course Focuses On Autonomous Robot Applications* Paper presented at 1999 Annual Conference, Charlotte, North Carolina
14. Xu, Y., & Li, H., & Jin, K. (2014, June), *Innovative STEM-preneur Learning Modules for a Freshman Robotic Engineering Class* Paper presented at 2014 ASEE Annual Conference & Exposition, Indianapolis, Indiana.
15. WPI 2017: https://www.wpi.edu/academics/departments/robotics-engineering.
16. Lawrence Tech 2017: https://www.ltu.edu/engineering/mechanical/bachelor-science-robotics-engineering.asp.
17. Gennert, M. A., & Padir, T. (2013, June), *Robotics as an Undergraduate Major: A Retrospective* Paper presented at 2013 ASEE Annual Conference & Exposition, Atlanta, Georgia.
18. Yousuf, A., & Lehman, W., & Mustafa, M. A., & Hayder, M. M. (2015, June), *Introducing Kinematics with Robot Operating System (ROS)* Paper presented at 2015 ASEE Annual Conference & Exposition, Seattle, Washington.
19. Yousuf, A., & Lehman, W., & Mustafa, M. A., & Hayder, M. M. (2016, June), *Low-Cost Robot Arms for the Robotic Operating System (ROS) and MoveIt* Paper presented at 2016 ASEE Annual Conference & Exposition, New Orleans, Louisiana.

20. Conrad, L. F., & Auerbach, J. L., & Howard, A. M. (2015, June), *The Impact of a Robotics Summer Undergraduate Research Experience on Increasing the Pipeline to Graduate School* Paper presented at 2015 ASEE Annual Conference & Exposition, Seattle, Washington.
21. Russell, Susan H., Mary P. Hancock, and James McCullough. "Benefits of undergraduate research experiences." *Science(Washington)* 316.5824 (2007): 548-549.
22. Zydney, Andrew L., et al. "Impact of undergraduate research experience in engineering." *Journal of Engineering Education* 91.2 (2002): 151-157.
23. Touretzky, David S. "Seven big ideas in robotics, and how to teach them." *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. ACM, 2012.
24. O'Kane, J. M. "A gentle introduction to ROS." Independently published, 2013.
25. Rosebrock, A. "Practical Python and OpenCV." *Miami: pyimageseach,* 2016.
26. Haykin, Simon S., ed. *Kalman filtering and neural networks*. New York: Wiley, 2001.
27. Hogg, Robert W., et al. "Algorithms and sensors for small robot path following." *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference*. Vol. 4., IEEE, 2002.
28. Prince, Michael. "Does active learning work? A review of the research." *Journal of engineering education,* 93.3, pp. 223–231, 2004.

Appendix A: Goals and Approach

Learning Activities and Objectives:

- Learn Linux, Ubuntu, ROS Setup.
- Learn Robotic Operating System (ROS) development environment.
- Learn how others have built ROS-enabled programs for robotics systems by recreating and
  extending several existing GitHub ROS examples.
- Be able to develop from scratch an ROS-enabled application in C++ or Python.

Stretch Goals:

- Understand the ROS simulation environment and create an ROS simulation.*
- Create a drone-based leader follower.*
- Document results.*

Learning Activities:

a) Building an OS and then the ROS environment on a Unix micro machine (Ubuntu). Platform will be a NUC-I7, Raspberry Pi, ODroid XU4, or equivalent. This will be provided to the student.
b) Interfacing with simulation software and GUI development in Python or C++. Demonstrate marker identification and other visual queues for use in autonomous navigation. Learn and use library functions in OpenCV, NumPY, FreeNect, and other enabling freeware.
c) Interfacing with hardware. This includes microprocessors to video devices, joysticks, pan and tilt mechanisms, sensors, etc. Contribute to the community via GitHub and collaborate with U.S. Military Academy (USMA) and other ongoing undergraduate research.
d) *Communicating with robotics and or other vehicles. Create ad hoc networks to pass information between platforms. Demonstrate cooperative movements, probably with TurtleBot to start.
e) *Developing an interface between the 3D robotics ArduPilot "Pixhawk" and the microprocessor and demonstrating autonomous flight. Demonstrate a leader follower drone and autonomous drone swarming capability. This may be demonstrated using a ground robot prior to aircraft use. Drones and ArduPilot autopilots will be provided to the student.

Evaluation:

20% bi-weekly meetings to assess progress; 20% final report; 20% part a; 20% part b; 20% part c, d,* and e.*

** Represents stretch goals. These items will act as bonuses if the student is able to achieve them.*