Proceedings of the ASME 2018 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2018
August 26-29, 2018, Quebec City, Quebec, Canada

IDETC2018

ENABLING GRADUATE ENGINEERING STUDENTS WITH PROFICIENCY IN MOBILE ROBOTICS

Adhiti Raman
Graduate Student
Department of Automotive Engineering
Clemson University
Email: adhitir@clemson.edu

Venkat Krovi
Michelin Endowed Chair Professor
Department of Automotive Engineering
Clemson University
Email: vkrovi@clemson.edu

Matthias Schmid
Research Assistant Professor
Department of Automotive Engineering
Clemson University
Email: schmidm@clemson.edu

ABSTRACT

In recent years, an aggressive expansion of research as well as commercialization efforts in autonomous vehicles can be witnessed. At the same time, many existing companies have expanded their portfolio to autonomous technologies as well (e.g. NVIDIA). This has created an already large need for autonomous-vehicle engineers who are not only proficient in single traditional engineering fields (e.g. mechanical) and old-school automotive studies, but who also have acquired the significantly different, interdisciplinary skillset for mobile robotics. Unlike students of computer science, mechanical engineering graduate students are hardly exposed to coding and robotic system integration in current traditional curricula. The new demands of the automotive industry require an automotive engineer who understands the science of autonomy as well as its impact on the design and implementation of autonomous vehicles, and is equipped with hands-on experience with the latest technology in the field.

We describe a unique education program that draws content from traditional courses on mobile-robotics as well as incorporates experiential learning by hands-on training in software, specifically addressing the skill gap in traditional automotive engineering education. Geared towards engineering students with no previous training in robotic system integration, and with only basic undergraduate understanding of programming languages, the teaching experiment employed an active learning approach to introduce numerous concepts as a host of hands-on exercises on multiple robotic platforms. Beginning with simple tutorials on networked communication to demonstrate the power of ROS, the course built up to complete control system design on a student-built RC car that can avoid obstacles and navigate a racecourse by performing SLAM.

A brief evaluation of the course exhibited good student performance in general with unique and creative approaches to the programming tasks in particular. Although employing different approaches, each student team was able to demonstrate comparable, efficient performance.

INTRODUCTION

Building high-performance, modular and verified Autonomous Vehicles (AV) requires solving several challenges in perception, (semi-)formal verification and control. Certain themes that AV research thrusts must focus on are:

Safe autonomy: Re-creating dangerous situations in a simulated Hardware-in-the-Loop (HIL) or Software-in-the-Loop (SIL) environment to perform risk-based safety analysis of AVs. Creating AV platforms that that use formally constrained

machine learning to learn from its experience and those of other AVs without departing from the safe behavior that got it certified (by formal of semi-formal verification techniques) as road worthy.

Agile autonomy: Developing computationally inexpensive control algorithms that enable agile maneuvering at high speeds.

Coordinated Autonomy: Using map-data, traffic conditions, road and visibility conditions for lightweight coordination to expand vehicle-centric safety bubble to a larger active network safe-zone.

All such tasks require high level networking between different AVs and sharing of computational expense of certain tasks with external and internal (online and offline) computing platforms. For this very reason, in the last few years Robotics Operating System (ROS) has emerged as a popular development framework in the autonomous vehicle industry. ROS was first introduced by Quigley et al. [1] in 2009 as an open-source framework for robot communication. While originally designed to enable large scale cross platform communication, it quickly become the de-facto standard in the robotics community. Now such knowledge and understanding of ROS is already a key requirement for the majority of open positions in the Autonomous Vehicle R&D industry.

The main objective behind the design of the course was to introduce mechanical/automotive and electronics engineers at Clemson University to the complexities and challenges of mobile robotics at a graduate level. The lack of previous exposure beyond a very basic knowledge of software programming made this a challenging objective. Incremental experiences and project based-learning had long been proven as a reliable format in teaching robotics to computer engineers [2].

In order to engage beginners quickly, we focused on a combined approach of in-depth coding tutorials and code implementation from existing repositories and projects. ROS is ideally suited for this approach by providing a framework that allows the communication of code in different languages over a common interface.

Incorporating existing ROS-enabled hardware platforms for robotics training has several apparent advantages: the TurtleBot robots by Willow Garage, for instance, is well suited for educational programs as it allows instructors to focus on the theoretical aspects of mobile robotics while at the same time allowing for physical learning experience. The use of the TurtleBot in education is well supported by literature [4]. Maas and Maehle compliment the object-oriented nature of robots such as TurtleBot and Bilibot [3], by highlighting their similarity to the concepts in object-oriented programming, hence making it ideal for introductory lessons. “Encapsulation”, for instance, allows the programmer to mask the complexity of the programming task by allowing only interfaced access to the hardware of the robot. The robots can be similarly controlled at a high level allowing the user to gain an intuitive understanding of the scope of the physical challenge before exposing the complex substructure i.e. the practical details of the programming challenge.

As the focus is on training graduate students, the envisioned learning curve was expected to be very steep. The students were expected to not only understand and control a ROS-enabled robot, but also to assemble their very own. The hardware platform for the task was based on the F1tenth competition by the University of Pennsylvania [5]. Here, the available Bill of Materials and the repository of tutorials allowed the students to engage quickly with the subject matter and add modifications of their own. The use of the F1tenth mobile base had another advantage as well. As mentioned previously, research on AVs has complex focus areas. This has also been historically limited to experimentation on expensive commercial vehicles, which require large teams, diverse skills and power hungry platforms. As this is outside the reach of many academic departments, they are constrained to teaching pieces of the overall autonomy picture, like motion planning and computer vision. An open-source platform such as this opens up new approaches to teaching autonomous perception, planning, control and coordination.

Teaching material for the course employed several references: *Advanced Mobile Robotics* by R. Seigwart [6] was mainly utilized for the theoretical knowledge imparted during class. The content for ROS tutorials was drawn from online resources such as GaiTech [7], the ROS Wiki page [8], MooreRobots [9] as well as online textbooks such as *Gentle Introduction to ROS* by author Jason M. O’Kane [10].

By the end of the course, students were expected to successfully provide answers to the following fundamental questions posed by Dudek and Jenkins in *Computational Principles of Mobile Robotics* [11] with competence:

- *Path planning:* Is it possible to get the robot from one configuration to another while remaining in C_{free} (the free space or *C-space* of the robot, i.e. the set of valid poses available to the robot).
- *Localization*: How can a robot determine its state if it has local measurements of C_{free}?
- *Perception or sensing*: How can the robot determine which parts of its environment are occupied?
- *Mapping*: How can the robot determine C_{free}, assuming it always knows where it is?

- *Simultaneous localization and mapping:* How can a robot determine its pose and C_{free} if it knows neither?

The framework for teaching was closely designed on the suggested format by Touretzky [4] which itself draws inspiration from curriculum design principles pertinent to abstract subjects as described in *Understanding by Design* by Wiggins and McTighe [12] and asks that the topic in consideration be taught in the following manner:

- *Motivate* by introducing an essential question, e.g. 'How do robots see?'
- *Demonstrate* the answer with an example of technology at work
- *Explain* how the technology works at a high level
- *Experiment* by allowing the students hand-on access to the technology through a project assignment
- *Review* the work through summarized reports to evaluate their understanding

This paper is organized as follows: in the following second section, we discuss the background, course format, and bi-weekly projects assigned. In the third section, we demonstrate how the relevance of the knowledge imparted helped shape the creativity displayed by the students in the final project assignment. Finally, we discuss some of the issues and challenges that were faced in implementing the subject matter and the long and short-term future of autonomy education.

BACKGROUND

Robot kits and small hardware platforms have increasingly been used for science and autonomy education since the 1990s. For example, Lego Robots have been popular in a wide variety of educational levels: from beginner lessons in high-schools to graduate level courses [13]. Menegatti and Moro [13] recommend the use of LEGO robots by reasoning that the ease of Lego's NXT programming allows the lecturers of graduate classes to focus on a conceptual and theoretical understanding of complex concepts (such as Monte Carlo Localization) without spending much time in the unromantic and technical aspect of robotics. However, it could be argued that in the industrial setting a technical proficiency may be equally important. Confronting potential hardware or software roadblocks and brainstorming solutions can provide a realistic expectation of challenges faced be an important part of a holistic robotics education.

A) Course Format

The Department of Automotive Engineering at Clemson University offers the class ***AuE893 - Autonomy, Science and Systems*** as a 4-credit course for graduate and post-graduate students in Automotive, Mechanical, Electrical and Electronics and Communication Engineering. There are no pre-requisites apart from a basic understanding of programming. The class size was limited to a maximum of 15 students to foster greater student-teacher interaction. In Spring '17, the course had 14 registered students (6 in the Master's program, 8 in the PhD program) who were split into 3 teams. While the majority of students was enrolled into the Automotive Engineering program, four students pursued degrees in Electrical, Electronics and Communications Engineering. The two ninety-minute in-class sessions per week were equally devoted to a lecture as well as demonstration of the current a problem statement by the instructor and TAs, and problem resolution combined with brainstorming in group discussions. Mini-projects and lab assignments were introduced weekly and were alternated with problem reviews and new concept introductions. While students were encouraged to solve the projects in cooperation with their teammates, they were still require to display their own work and highlight each team member's contribution through citations. The overall course was broadly split into three modules:

1) Module 1: Introduction (5 weeks)

i. Introduction to Python, C++ and working on a Linux operating system.
ii. An introduction to GitHub and coding as a team.
iii. An introduction ROS – working with catkin, creating packages, writing publishers and subscribers, launch files, Turtlesim, Gazebo, RViz and simple demonstrations with a Turtlebot.

2) Module 2: Coding for Autonomy (6 weeks)

Covering the fundamental problems in mobile robotics with a Turtlebot and Gazebo

i. Obstacle avoidance and PID control
ii. Sensing the environment, localization and path planning
iii. Mapping and SLAM with sensor fusion

3) Module 3: Capstone Project (5 weeks)

Building your own ROS-enabled autonomous race car. Each team was given all the parts and a variety of sensors with the freedom to build their own navigation controller with the tools on hand.

Five laboratory assingments or "mini projects" were spread over the 3 modules. Brief descriptions are given below

Lab	Topic	Learning Goal
1	Getting started	Review of programming, introduction to Ubuntu and GitHub platforms.
2	Basics of ROS	Create packages, publish, subscribe and make launch files. Using TurtleSim, Gazebo.

3	Sensors	Working with simulated sensors, editing Gazebo worlds and mapping the worlds.
4	Obstacle Avoidance	Move-to-goal, obstacle avoidance, path planning.
5	Navigation	Using PID control to autonomously navigate a racetrack with obstacles.

B) ROS-Enabled Robotic Platforms

Two primary robotic platforms were used in class. The Turtlebot by Willow Garage has long been a popular choice for beginner robotic courses in ROS development [4,14,15]. The F1tenth platform was used for the first time as part of an educational course by us.

Weeks 1-11: For two-thirds of the course we used the newly released **Turtlebot-2** which comes on a Kobuki base on a two wheeled differential drive system with cliff sensors, wheel encoders, bumper sensors, an Orrbec Xtion Pro RGB-d camera and an ASUS laptop running Ubuntu 14.04 with ROS Indigo. Most of the project coding and implementation was done on a Gazebo simulation and tested live in class on a real Turtlebot. There were some challenges faced while using this platform. At the time of the Turtlebot2 release, the popular Microsoft Kinect sensor had been taken out of circulation and the last-minute replacement provided – the Orrbec Xtion Pro -- had not been sufficiently play tested. As a result, a working ROS driver for the camera was in development and was not available during the time of course. A significant amount of time was spent attempting to modify the existing OpenNI2 driver with little success. This was an excellent real-world example of software and hardware roadblocks often faced in the field of robotics and culminated in an interesting discussion of alternative approaches to project challenge at hand. With the ability of depth perception taken out of the running for the nonce, the students explored using the bumper sensor in their obstacle avoidance code.

Weeks 12-16: In the final five weeks of the course, the robotic platform used was the **F1tenth car** developed by the University of Pennsylvania [5]. The F1tenth Competition is an annual contest hosted by UPenn every year and in it, student teams compete to build and race autonomous RC vehicles. The website for the competition is a well-maintained repository of short guiding tutorials and part recommendations for the hardware build of a ROS enabled robotic car. Although the online repository and bill of materials were outdated at the time, with many components no longer in circulation, we nonetheless found this to be an excellent resource base for use in coursework. The RC car built by the students in the course used a Traxxas Slash 2WD Short Course Truck with laser cut platforms and 3D printed fixtures for housing the different components and sensors. The computing platform for running the Linux operating system and the overlying ROS framework was the NVIDIA TK1 or TX1. The accompanying sensors were a Hokuyo UST-10LX LiDAR, a choice of Structure Senor depth camera or ZED RGB-d stereo camera, and SparkFun's 9DoF Razor IMU. A Teensy 3.2 microcontroller board programmable with Arduino was employed to control the ESC and steering servos with serial commands through ROS. Peripherals included a Ubiquiti PicoStation for Wireless access, USB hubs and a generic power bank with 12V and 20V supplies.

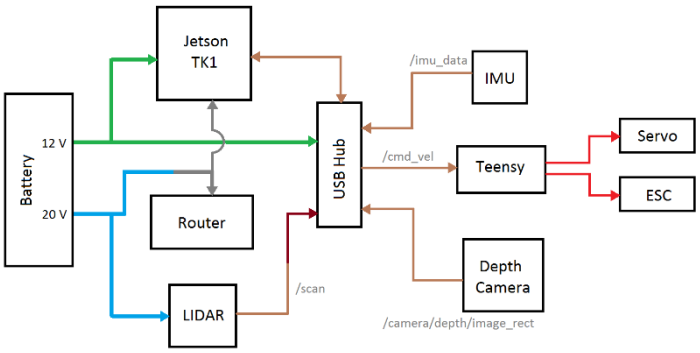

Fig 1: System architecture and connections of the parts in the F1tenth car.

C) Capstone Project

The five weeks of effort put into the capstone project by the respective teams was tested in a final examination held at the end of the semester. The examination included a 20 minute presentation on the build and working on the car followed by a live demonstration on a race track. The students were expected to submit a report on the project in the style of a research thesis.

The objectives of the final project were three-fold: **a)** apply the concepts learned in class (path planning, mapping, control) on a robot built from scratch. **b)** work in a team, share code and ideas to create something novel together. **c)** understand the challenges in research and the limitations of autonomous vehicles in a real world scenario. The three teams were tasked with a achieving a 3-point agenda:

1. Achieve autonomous navigation around an arbitrary racetrack with the sensors at hand (depth camera, RGBD camera, LiDAR, IMU).
2. Perform mapping and localization while remote streaming the data
3. Obtaining quality obstacle avoidance at high speeds and low speeds.

The final demonstration was held on a racetrack with movable obstacles.

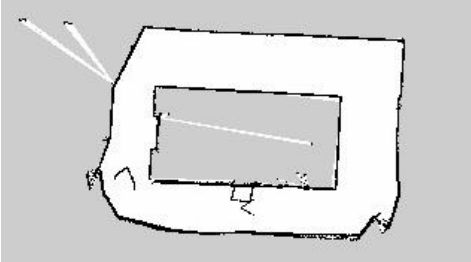

Fig 2: Hector map of the final racetrack taken by one of the cars

LEARNING OUTCOMES

Each of the teams had unique and creative approaches to the problem statements. The primary languages supported by ROS are Python and C++, with significant (and growing) ROS support available through the MATLAB Robotics Systems Toolbox. Two teams chose to build their controller in Python and one team chose MATLAB. The most significant divergence in tactics employed to solve a given problem was demonstrated in their respective solutions to the problem steering and velocity control during racecourse navigation using the LiDAR laser scan output.

Team A: Pure Pursuit

Pure pursuit works from the idea of determining a goal point in the vehicle frame, and then calculating the necessary steering angle and velocity inputs to reach this goal point. This pure pursuit problem is broken down into three sections: calculating the goal point, calculating the required steering inputs to get to the goal, and calculating the appropriate velocity commands based on the goal point.

Goal Point Determination: The formulation of a goal point was slightly unconventional. Information of the surroundings is local information taken from the LiDAR sensor, and the goal point selected was the centroid of all the data points.

$$\begin{bmatrix} x_t \\ y_t \end{bmatrix} = r_i \begin{bmatrix} \cos(\theta_{start} + i\Delta\theta) \\ \sin(\theta_{start} + i\Delta\theta) \end{bmatrix}$$

Steer Control: For simplicity, the Ackerman steered vehicle was modelled as an equivalent bicycle. Using the goal point, the look ahead distance was calculated by taking the hypotenuse of the triangle, which is formed by the goal point, and the vehicles heading and position. The look ahead distance is then used to calculate the curvature of the desired trajectory (arc) from the rear axle to the goal point. By incorporating the wheelbase of the vehicle, the desired steering angle ϕ was calculated. Instead of using this steering angle directly, a gain factor k_{st} was introduced to tune the responsiveness of the steering. A steering bias b_{st} was introduced to tune the steering trim and compensate for misaligned wheels.

$$Angulartwist = k_{st}\phi + b_{st}$$

Velocity Control: Conventional pure pursuit determines a goal point as a function of the vehicle velocity. In this implementation, the velocity was determined from the goal point. Not only does the goal point shift laterally as the vehicle navigates the track, but it also shifts longitudinally. In the sections of the track where there is a clear path far from the vehicle, the goal point is further away thus the look ahead distance is greater. When the vehicle could calculate a trajectory to a greater distance, the vehicle would go faster. Conversely, if the vehicle could only calculate the trajectory which is very close, the vehicle would slow down. Using this methodology, the velocity was calibrated to a given look ahead distance linearly.

Team B: "The Potential Field"

Steering control: The potential field is found by finding the sector that has the largest average distance. The angle between the target sector and the centerline α_{pf} is the first component of the input error to the steering PD controller.

For collision avoidance and centering the PD control algorithm was extended further to incorporate two more error terms. The first term α_{avoid} was calculated as a function of the sum of the distance along x-axis of each laser point. To ensure that objects not lying directly in the path of the robot do not affect the performance of the obstacle avoidance code, the laser scan data was trimmed down to take the shape of the "M" shaped path in the center. (|x|<=0.2)

Finally, for centering during cornering, the laser scan was divided into four sectors on right and four sectors on the left side. The average distances along X axis and Y axis are calculated based on the average range of each sector. The distances on each side are then merged into a straight line separately by using line fit. With the two fit lines, the distance D between the vehicle center O and the track center can be estimated and used to calculate the final error, α_{cen}.

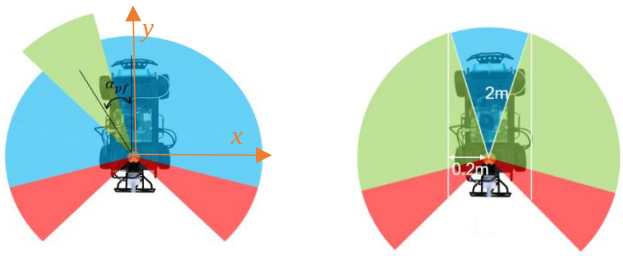

Figure 3: (Left) Finding the potential field. (Right) Path trimming for collision avoidance

Speed control: A simple P controller was used to control the vehicle speed. The speed controller takes input from keyboard. The up and down arrow key will adjust the lower speed limit and the controller will calculate the vehicle speed in proportion to the minimum distance along x axis. The vehicle will stop if the

space key is pressed or the average range of the middle sector r_avg (middle) is below the minimum allowed threshold r_(avg-critical). The functionality of the speed controller can be described by the following equation:

Team C: Overlapping Sectors

Steering control: The designed heading was chosen as the centerline of the section where the LiDAR data had the largest value. The error used to implement a proportional–derivative (PD) controller was defined as the error between the current heading section and the desired heading section.

To handle the width of the vehicle, the size of the section was set as 45 degrees. The step size of LiDAR scan was defined as 5 degrees. Therefore, the overlapped size of two adjacent sections was 40 degrees. The weight of each section was defined as the sum all the feedback values of LiDAR beams. The section with has maximum weight was chosen as the desired heading section, and the angle of the its center line is the desired heading angle.

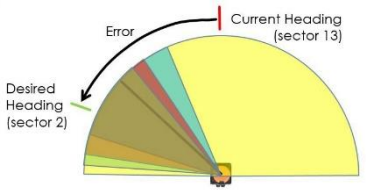

Figure 4: Overlapped Section

Speed control: The Lidar based speed is also designed for reach the highest speed in turning. To achieve this the vehicle needed to decelerate when approaching the corner, and reach a smaller speed. Moreover, the shape of the corner determined the speed at the corner. The vehicle should be accelerated as soon as possible when it departures the corner. A proportional controller was used to achieve this goal. The input of this controller is the same error that used for steering control. The vehicle always runs in maximum speed in straight line and no obstacle in the front (error = 0), keep decelerating when it approaches the corner until it near to the center of the corner. After passing the center of the corner, it will accelerate if the LiDAR sees a free space in the front.

Implementations of the following control systems can be found on the course YouTube channel [17]

DISCUSSION

By the end of the term, the standalone car models that had been built and designed by each of the student teams bespoke controller models for navigation, which were comparably efficient. There were logistical problems and delays faced in sourcing the components required. Unexpected problems with hardware and software compatibility that arose were dealt with by the students' ingenuity, resourcefulness and teamwork.

There are many important theoretical concepts that go beyond the scope of the current curriculum that would need to be addressed in future revisions such as motion planning using A* and Dijkstra algorithms. Further implementations on advanced algorithm design such as achieving robot localizations and goal point determination and path planning on a pre-existing Hector SLAM [18] maps using only scan-matching could be attempted. Other robotic platforms that may enhance and supplement the future course format were built and deployed by student developers. Two of the most successful robotic platforms for this purpose that may be implemented in future classwork are the NVIDIA Jetbot and Q-bot by Quanser. The JetBot ships with an NVIDIA Jetson TX1 on a differential drive platform with an Arduino Mega for ROS-serial control of the driving servos and Ultrasound sensors for navigation. The Q-bot by Quanser is operated on a Kobuki base similar to the Turtlebot2 and ships with a Microsoft Kinect. It is not a ROS-enabled robot but contains a useful MATLAB interface which serves as an excellent introduction to Simulink and MATLAB Real-time Workshop for robotic applications.

All of the above can be further expanded upon into two semester course module with the second phase focused on advanced algorithms, dynamic control, incorporating machine learning and control for object recognition and path planning incorporated within a ROS-wrapper using the NVIDIA Jetbot and the F1tenth.

For a student majoring in engineering today, targeted courses in mobile robot autonomy could pave way for the establishment of a "Department of Autonomy" focused on teaching perception, planning, control, coordination for transportation, industrial automation and energy systems set in the socio-economic context of safety, fairness and affordability. There is already a rising trend favoring autonomy education in the context of intelligent transportation systems and industrial robots. In the future we plan to expand upon these ideas and develop new courses a) combining machine learning and control and b) a hands on course on "agile" autonomy, where various aspects on autonomy are tested in an arena that is made up of race tracks, obstacles, barriers and visual markers.

Beyond the classroom, the fleet of platforms can be employed to enable scaled research and traffic modelling such as platooning and for research in autonomous driving in general.

ACKNOWLEDGEMENT

The authors thank the students of the course for their enthusiastic participation and positive attitude when faced with challenges in this experimental course. We would also like to thank the doctoral students, Howard Brand and Yi Chen, without whose help and contribution this course would not have been the success it was. Finally, we would like to give our thanks to the

members of the Clemson IT services who were vital to the smooth running of the class.

REFERENCES

[1] Quigley M. et al., “ROS: an open-source Robot Operating System,” Icra, vol. 3, no. Figure 1, p. 5, 2009.

[2] Nourdine, A., “Teaching fundamentals of robotics to computer scientists” In Computer Applications in Engineering Education Volume 19, Issue 3, pages 615–620, September 2011

[3] Maas, R. and Maehle, E. "An Easy to Use Framework for Educational Robots," ROBOTIK 2012; 7th German Conference on Robotics, Munich, Germany, 2012, pp. 1-5.

[4] Touretzky, D. S., “Seven big ideas in robotics, and how to teach them,” Proc. 43rd ACM Tech. Symp. Comput. Sci. Educ. - SIGCSE ’12, p. 39, 2012.

[5] F1Tenth Competition, http://f1tenth.org/

[6] Seigwart, R. and Nourbakhsh, I.R., 2004, *Advanced Mobile Robotics*, The MIT Press, Cambridge, Massachusetts

[7] Koubaa A., Lee, Z., 2016, “Gaitech EDU” http://edu.gaitech.hk/ros/ros-tutorials.html

[8] ROS Wiki, http://wiki.ros.org/ROS/Tutorials

[9] MooreRobots, http://moorerobots.com/blog

[10] Kane, J. M. O., “A Gentle Introduction to ROS”, 2014,

[11] Dudek, G. and Jenkin, M., 2010, *Computational Principles of Mobile Robotics*, Cambridge University Press

[12] Wiggins, G. and McTighe, J., 2005, *Understaning by Design*, Assn. for Supervision & Curriculum Development

[13] Menegatti, E. and Moro, M., “Educational Robotics from high-school to Master of Science,” Proc. SIMPAR 2010 Work. Intl. Conf. Simulation, Model. Program. Auton. Robot., no. March, pp. 639–648, 2010.

[14] Riek, J.D., “Embodied computation: An active-learning approach to mobile robotics education,” IEEE Trans. Educ., vol. 56, no. 1, pp. 67–72, 2013.

[15] Ruzzenente, M., Koo, M., Nielsen, K., Grespan, L., and Fiorini, P., “A Review of Robotics Kits for Tertiary Education,” Proc. 3rd Int. Work. Teach. Robot. Teach. with Robot. Integr. Robot. Sch. Curric., pp. 153–162, 2012.

[16] Krovi, V., “Autonomy Science and Systems”, http://cecas.clemson.edu/aue893_autonomy

[17] YouTube Channel: https://www.youtube.com/channel/UCy9IYh5KCgeei4LzDg9hmHA

[18] Kohlbrecher, Stefan, et al. "Hector open source modules for autonomous mapping and navigation with rescue robots." Robot Soccer World Cup. Springer, Berlin, Heidelberg, 2013.