A Hands-on Middle-School Robotics Software Program at MIT

Sabina Chen, Andrew Fishberg, Eyassu Shimelis, Joel Grimm, Scott van Broekhoven, Robert Shin, Sertac Karaman
Massachusetts Institute of Technology[1],
{sabinach, fishberg}@mit.edu, {eyassu.shimelis, grimm, vanbroekhoven, shin}@ll.mit.edu, sertac@mit.edu

***Abstract* - Robotics competitions at the high school level attract a large number of students across the world. However, there is little emphasis on leveraging robotics to get middle school students excited about pursuing STEM education. In this paper, we describe a new program that targets middle school students in a local, four-week setting at the Massachusetts Institute of Technology (MIT). It aims to excite students by teaching the very basics of computer vision and robotics. The students program mini car-like robots, equipped with state-of-the-art computers, to navigate autonomously in a mock race track. We describe the hardware and software infrastructure that enables the program, the details of our curriculum, and the results of a short assessment. In addition, we describe four short programs, as well as a session where we teach high school teachers how to teach similar courses at their schools to their own students. The self-assessment indicates that the students feel more confident in programming and robotics after leaving the program, which we hope will enable them to pursue STEM education and robotics initiatives at school.**

Index Terms – Middle School Outreach, Project-based Learning, Robotics, Computer Science.

INTRODUCTION

Robotics is a rapidly-emerging industry with a potential to impact many businesses. It is conceivable that many students who are in middle school and high school education today will be working on engineering systems that embrace robotics and software. Hence, there is a need for getting students in grade school excited about robotics early. Teaching them the main concepts of engineering will enable them to pursue STEM education in the future.

To address this need, in 2016 we started the Beaver Works Summer Institute (BWSI), a four-week residential STEM program for high school seniors focused on teaching emerging technologies [1]. Meanwhile, there continues to be a rapidly-growing number of high school programs dedicated to robotics. Some of the most popular and notable programs include the FIRST Robotics Program [2], BEST Robotics Program [3], and the ZERO Robotics Program [4]-[7]. These programs all share common threads of understanding the value of hands-on active learning with robotics [8-10].

While the BWSI program has grown substantially each summer, we continue to learn and improve each year. Two important lessons emerged from the first four years of the high school course: the importance of reaching students before their senior year of high school and the value of further reducing the cost of our platform to promote scalability.

Our first lesson had us considering how to reach younger students. As other outreach enthusiasts note, timing is critical when promoting students pursue an STEM discipline [11]. Students face pragmatic difficulties if they try to hard pivot into a STEM field by the time they reach the BWSI program as a high school senior. This is especially true if they chose to pursue a less rigorous math track through high school. While some of the aforementioned high school robotics programs do reach middle school students, and long running middle school robotics programs do exist [12], there is an identified dearth of access and curriculum materials, especially for middle school instructors [13].

Our second lesson had us considering how to reduce the cost of our platform. While our high school robot, was comparably priced to other robotics programs, approximately $5000, many external collaborators expressed an interest in a similarly capable, cheaper platform. The mobile robotics community already produces lots of cheap, student-ready platforms [14]. Some notable examples include Legos robotics kits [15-19], iRobot kits [20-22], or laptops with wheels [23]. One recent platform, Duckietown, deserves special shout-out, as it shares many similar design philosophies about flexibility, course content, and software stack [24-25]. That being said, these reduced prices often

[1] DISTRIBUTION STATEMENT A. Approved for pubic release: distribution is unlimited. This material is based upon work supported under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. Air Force.

978-1-7281-7520-1/20/$31.00 ©2020 IEEE

come with significantly reduced capabilities, especially in the movement speed.

In this paper, we describe a new middle school program that teaches robotics software in the context of programming self-driving mini racing cars, which we have developed in-house at the Massachusetts Institute of Technology (MIT). This new program is a four-week summer program at MIT, which we ran in Summer 2019 for the first time. The course focused on teaching students fairly advanced concepts in programming, computer vision, algorithmic robotics, and robot software. The program was enabled by a new hardware and software infrastructure system that substantially simplified the programming for robot navigation using a camera sensor. We present this enabling hardware and software infrastructure, the curriculum, and the results of a self-assessment questionnaire. We also ran a training program for teachers to enable them to teach the same material at their own schools. We present the results of a questionnaire with the teachers regarding their learning and their assessment of the program's value for their school. All course materials, hardware designs, software and exercises are freely available to the community at our website (https://mit-bwsi-racecar-ms.github.io/website/).

This paper is organized as follows: First, we present the hardware and software infrastructure that enables our program. We discuss a number of elements, particularly the utilization of Jupyter notebooks, that make the programming of the vehicles far easier, for instance, when compared to what the environment we reported in our earlier work with high school students [1]. Next, we describe the details of the program, specifically outlining the various components and lectures. Then, we present results of self-assessment surveys for both the students who were enrolled in our program and the teachers we trained. Finally, we conclude the paper with remarks.

HARDWARE & SOFTWARE INFRASTRUCTURE

In this section, we describe the hardware platform and the software infrastructure, which we have developed in-house to enable this program.

a) Hardware Platform: The new hardware platform that we utilize in this middle school program is based on our previous hardware platform called the MIT RACECAR (https://racecar.mit.edu). The RACECAR platform includes a state-of-the-art embedded computer (NVIDIA Jetson Tegra X2) together with a stereo camera (by ZED) and a planar laser range finder (by Hokuyo). The RACECAR platform typically costs around $5,000. We have built the MIT RACECAR for MIT's undergraduate robotics course, and have been utilizing it for a high-school program [1] describes the details of the RACECAR platform as well as the high-school program.

The "RACECAR Model-N," which we will refer to as the RACECAR-N from here on, is a new platform designed, for this middle school program, to be much easier to use and much more affordable. The platform is powered by an NVIDIA Jetson Nano embedded computer, which boosts a 128-core Maxwell-architecture GPU, a quad-core ARM A57 CPU, and 4GB of memory. The platform also includes an Intel RealSense 435i camera and depth sensor, which provides 1920x1080-resolution camera images at 30 frames per second and 1280x720-resolution depth images at 90 frames per second together with 6-degree-of-freedom inertial measurements. The platform also contains a YDLIDAR unit with 10m range and 360 field of view. The computing and sensing system is mounted on a 1/14th-scale RC plat- form, the "Exceed RC 1/14 Tacon Thriller Short Course Truck." This chassis includes an electric drive motor to drive the wheels and an electric servo motor to steer the front wheels. We added a Pololu 6-channel servo controller to control these motors. The mounting plate is laser cut from Delrin Polyoxymethylene material. The design and the laser cutter files are available for download on our website. The fully-assembled platform is shown in Figure 1.

Figure 1. RACECAR-N with most important components highlighted.

In addition to the RACECAR-N hardware shown in the figure, we have also added a portable 7" monitor that can be connected to the NVIDIA Jetson Nano embedded computer via the HDMI interface, a Logitech K400 Plus wireless keyboard/trackpad, and the TP-Link AC1750 Smart WiFi Router. Using these devices, the students can easily run and debug the software on the RACECAR-N.

b) Software Infrastructure: The NVIDIA Jetson Nano embedded computer runs on the Ubuntu Linux operating system provided by NVIDIA's Linux 4 Tegra Jetpack. The software infrastructure includes the Robot Operating System (ROS), which is one of the most widely used robotics middleware. This software infrastructure is widely-utilized in many robotics research and development projects [26], and it has formed the backbone of our undergraduate research at MIT, as well as the high school summer program at MIT [1]. However, arguably, this software infrastructure is difficult to use as-is for many reasons. Firstly, the development, compilation and execution of the software requires the understanding of a complex architecture involving binaries, libraries, and various ROS tools. Secondly, most of the process of compilation and execution of software requires command-line tools. In many instances, such tools must be invoked from another terminal by connecting to the car through Wi-Fi or Ethernet.

In order to simply software development towards making it accessible for middle school students, we developed additional software infrastructure, which allows students to develop their own software via Jupyter notebooks, which is easily accessible from a web browser. For this purpose, we have installed a Jupyter server on the NVIDIA Jetson Nano embedded computer on the RACECAR-N platform. We have also made the ROS tools and the OpenCV tools accessible from the Jupyter environment. In this new environment, the students can connect to the car via a Jupyter notebook running on their laptops, or better yet, they can simply use the wireless keyboard/trackpad and the 7" screen connected to the embedded computer on the car. In both cases, the Jupyter notebook runs on a web browser, where the students can develop and execute software. The ROS and OpenCV functionality is provided to them via simple functions, e.g., drive the motor at a given speed or steer the front wheels at a given angle. This new infrastructure eliminates the need to interface with complex software architecture.

We observed that the new software infrastructure simplifies the software development process for robotics tremendously, allowing students to focus on developing and implementing core algorithms on the robot in a matter of minutes.

CURRICULUM

In this section, we describe the salient features of the program and its implementation.

	Spring 2019	Summer 2019	Fall 2019	January 2020
Name	Racecar Middle School (Roxbury)	Racecar Middle School (BWSI)	RMS Crash Course (All Girls)	RMS Educator's Training (Local MS / HS Teachers)
Length	4 weeks	1 month	8 weeks	4 weeks
Times	Saturdays 9:45am - 2pm	Monday - Friday 9am - 3pm	Saturdays 11am - 2pm	Saturdays 9am - 5pm
Typical Schedule	9:45am Warmup 10am Lecture 12pm Lunch 12:15pm Hands-on Exercise 2pm End of Class	9am Lecture + Lab 11am Technical Seminar 12pm Lunch 1pm Lecture + Lab 2:30pm Kahoot! 3pm End of Class	11am Lecture 12:15pm Lunch 1pm Lab 2pm End of Class	9am Lecture + Lab 12pm Lunch 1pm Lecture + Lab 5pm End of Class
Topic Overview	Weeks 1-2 Python Weeks 3-4 RACECAR	Week 1 Python Weeks 2-3 OpenCV Week 4 RACECAR	Weeks 1-6 Python Week 7 OpenCV Week 8 RACECAR	Week 1 Build Week 2 Python Week 3 OpenCV Week 4 RACECAR
# of Students	20	24	19	22
Staff	1 Instructor 5 TAs	1 Instructor 5 TAs	1 Instructor 2-3 Lincoln Lab Volunteers	2 Instructors
Hours	4 hrs/day * 4 days = 16 hours	5 hrs/day * 19 days = 95 hours	2.25 hrs/day * 8 days = 18 hours	7 hrs/day * 4 days = 28 hours
Location	Roxbury Innovation Center	MIT	MIT	MIT

Figure 2. The details of the four instances of the program.

a) Program: The program was taught in four instances. The first instance was during the Spring of 2019, targeting local middle school students from inner city Boston. The program ran on four Saturdays from 9:45AM to 2PM. The second was during the Summer of 2019, targeting local middle school students. This instance was organized into a summer school of 4 weeks, Monday through Friday, from 9AM to 3PM. The third instance was an all-girls class, run during Fall of 2019 during the weekends, on Saturdays from 11AM to 2PM for 8 weeks. Finally, the fourth instance targeted high school and middle school teachers in January 2020, and ran on Saturdays for 4 weeks, from 9AM to 5PM. This information is summarized in Figure 2 along with information on a typical agenda, the topic overview, the number of students, and the full-time staff. An estimate of number of hours of technical instruction and technical exercises is also reported in the figure.

b) Technical Content: The technical content differs slightly between the four different instances of the program, as seen in "Topic Overview" from Figure 2. In this section, we explain the technical content for the Summer 2019 instance of the program, which covers the most amount of technical material. The initial Spring 2019 session, as a shorter class, covered a subset of the material later taught during the summer. During the Fall 2019 all-girls offering, we left out a few of the lectures from the modules listed in the topics due to time constraints. In the January 2020 instance, taught to middle school and high school teachers, we added in an entire day to build the cars from their off-the-shelf parts. Building the cars is not included in the general middle-school program curriculum due to logistical and time difficulties. However, this module was added in for the teachers to enable them to have a more grounded understanding of the racecar components that they will be teaching and building themselves in the classroom.

The technical content is split into three modules: (i) essentials of programming with Python, (ii) fundamentals of computer vision with OpenCV, and (iii) basics of autonomous robotics software. In addition, after completing these modules, the program features a "final challenge," in which the students build on the materials that they have learned, in order to develop software for a fully-autonomous vehicle to navigate through a mock racing course.

Each module is divided into several lectures. Each lecture includes a slide presentation as well as an associated Jupyter notebook for interactive exercises. In many Jupyter notebook exercises, the students "fill in the blanks" after understanding the foundational concepts. Conceptually, the students fill in missing code pieces to solve small problems that are described to them in the Jupyter notebooks; the concepts required to solve these problems are provided in the slides presented in the lectures. As the modules advance, the students need to fill in larger amounts of software. In the final challenge, the students build most of the software system themselves. All of the slide presentations and the Jupyter notebooks are available on the course website (https://mit-bwsi-racecar-ms.github.io/website/).

The Python module teaches the basics of programming, including storage concepts (e.g. data types and variables), conditional statements, functions, loops, and also more complex data structures, such as tuples and dictionaries. It also has an advanced lecture on object-oriented programming. These topics are already fairly advanced material at the middle school level. Therefore, making the Jupyter notebooks explanatory, accessible, and full of interactive exercises, enables us to go over this material with the students in a more in-depth, but digestible manner. The exercises are supported with slide presentations which give an overview of the concepts to be applied.

In addition, we have prepared a second part on Python programming that focuses on building basic games with Python, which we have observed to especially attract students to the material. Specifically, we have prepared lectures to build a tic-tac-toe game, a hangman game, and a cube runner. These exercises can be completed with the help of instructors, and immerse the students into the programming exercises, since the students can explicitly see their software execute in an engaging way, allowing them to better grasp the essentials of Python programming.

The fundamentals of computer vision with OpenCV module start from the very basics of image manipulation and gradually introduces students to more useful, complex computer vision tasks, such as object recognition. The module starts with a lecture on displaying shapes and colors on the screen, as well as loading up images from the computer disk or the camera. The next two lectures introduce students to the basics of colors (e.g. generating colors through combining red, green and blue pixel values) and color spaces (e.g. hue, saturation and value). The following two lectures then give an overview of color masking, teaching the students how to extract shapes of a certain color from the camera image by manipulating color spaces. Finally, the module concludes with lectures that enable students to identify objects within an image by applying more complex topics, such as extracting contours from masks, computing the locations and sizes of masks (e.g., bounding boxes), detecting features, and detecting edges in images. This module is presented in small immersive exercises that require students to complete code snippets to correctly execute the program.

The final module on autonomous robotics software teaches the basics of algorithms for autonomous navigation. It includes five separate lectures, grouped into three parts: (i) cone detection and cone following, (ii) line detection and line following, and (iii) sign detection. The first part includes two lectures on cone detection and cone following. The goal is to program the behavior for the car to detect a cone, drive the car towards the cone, and then park the car at a certain distance in front of the cone. The software uses color masking (which the students learned in the OpenCV module) to detect and isolate the cone in the image, and then to create a bounding box around it. Then, based on the location and size of the bounding box, the software must detect the relative orientation of the cone with respect to the car and the distance between the cone and the car. Finally, the software must determine the steering angle necessary to drive the car towards the cone, and stop the car when the distance between the cone and the car reaches the desired value. The second part focuses on line detection and following. In this lecture, the software must detect a line on the ground and steer the car in order to follow the line based on a specified “look-ahead” distance. Line following is a challenging computer vision task because students must develop their software not only to detect the line, but also to understand which direction it must curve towards. Line following directly builds upon the concepts learned in OpenCV and cone following, but with a slight twist in application. The third part is a single lecture on detecting signs using basic feature detection algorithms, such as SIFT, SURF, and ORB, available in OpenCV.

The final challenge includes a complex map that has different color lines laid on the ground. Students use the concepts learned in the past three modules: Python, OpenCV, and Robotics to program their racecar to autonomously navigate the track. By just following one color, the students can successfully navigate the track by using a simple line follower. Shortcuts are also available for students who wish to apply the more complex topics taught in class. Yellow colored lines, orange cones, as well as one-way signs act as shortcuts. The more in-depth the students are able to understand the concepts taught in class and apply it to the final challenge, the faster and more accurately their car will be able to finish the track.

A sample track for two teams is shown in Figure 3. The final challenge is split into two parts: (i) individual time trials and (ii) a tournament style competition, with two teams competing on the track at a time. The tournament teams are initialized with the results of the individual time trials.

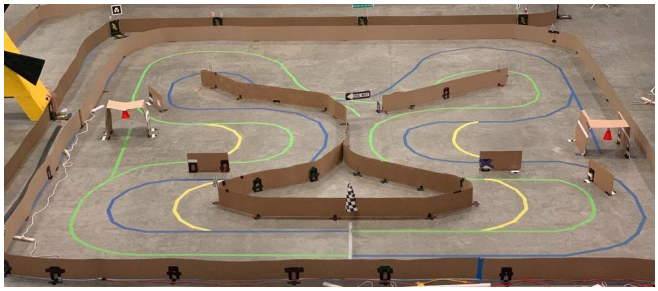

Figure 3. An example racing track. The start locations for the two teams are shown at the front center by the checkerboard flag. One team follows the green line and the other team follows the blue line on the course. Following the yellow tape, cones, and one-way signs create shortcuts. Other racing courses can be created similarly.

The three main modules, Python, OpenCV, and Robotics, are taught in one week each. The last week is dedicated to the final racing challenge. The students work in teams of two to develop software to navigate through the course fully autonomously during this time. Recall that the total length of the Summer 2019 program is four weeks.

ASSESSMENT

We present a preliminary assessment of this program via a set of self-assessment questionnaires. In this section, we present the results of the self-assessment questionnaires for the Summer 2019 program, and the January 2020 teacher’s training program.

a) Self-assessment questionnaires for students: With the students, we have attempted to assess their comfort level with (i) General Programming, (ii) Python, (iii) OpenCV, (iv) Robotics. For this purpose, we directed the questions to ask: “How would you rate your comfortability with ...?” The choices ranged from 1 to 5, with 1 being ”not comfortable”

and 5 being ”very comfortable”. There were 24 students total in the Summer 2019 session who took the same questionnaire on the First and Last day of class. The results are presented in Figures 4-7.

In Figure 4, we observe that the students come to the class with some knowledge of programming, even though most of them rate themselves as neutral or not very comfortable. It is worth noting that no student chose “not comfortable” (score of 1). Typically, even in middle school, we find that there is some emphasis of some kind of programming, which makes the students familiar with concepts and gives them some comfort level. When students leave the program, we observe that most of them rate themselves as comfortable with programming. We notice that the number of students who rate themselves very comfortable (score of 5) increases, but not by a large degree. However, we notice that many students rate themselves comfortable or neutral.

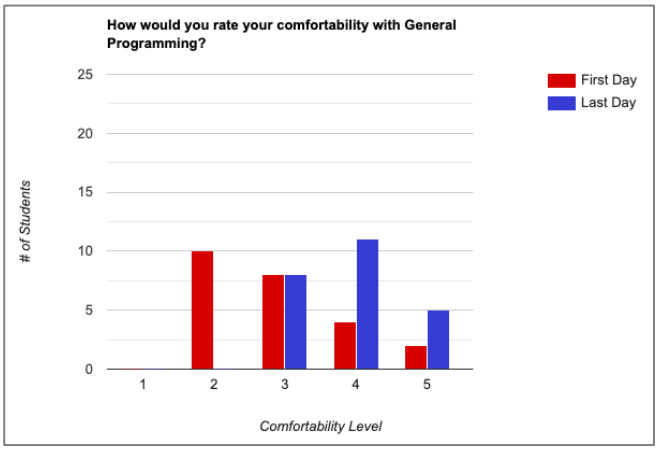

Figure 4. The student self-assessment questionnaire results for General Programming.

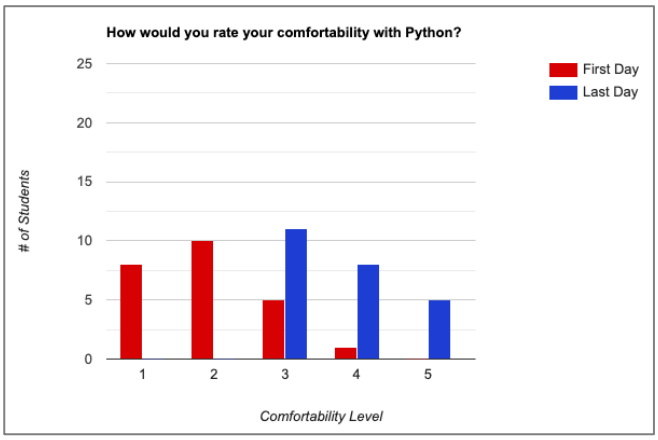

Figure 5. The student self-assessment questionnaire results for Python.

In Figure 5, we observe that the students do not know Python. Contrast this with their starting point in comfort level with general programming shown in Figure 4. The students become familiar with various ways of programming before coming to our program. However, only a few of them feel comfortable with Python. In fact, almost all students rate themselves as neutral or not comfortable with Python on the first day. We observe that the students leave our program with all students rating themselves at least neutral or comfortable with Python, with the majority feeling comfortable.

In Figure 6, we asked the students about how familiar they are with OpenCV. OpenCV is an advanced computer vision library that most computer science students are not exposed to until the college-level. Furthermore, most middle school students have never worked with real images in their computer science classes or relevant extracurriculars. Therefore, it is not very surprising that almost all students had never heard of OpenCV, with 22 out of the 24 students giving a rating of 1 (not comfortable) on the first day. However, as further shown in Figure 6, the students leave our program with relatively higher comfortability in OpenCV.

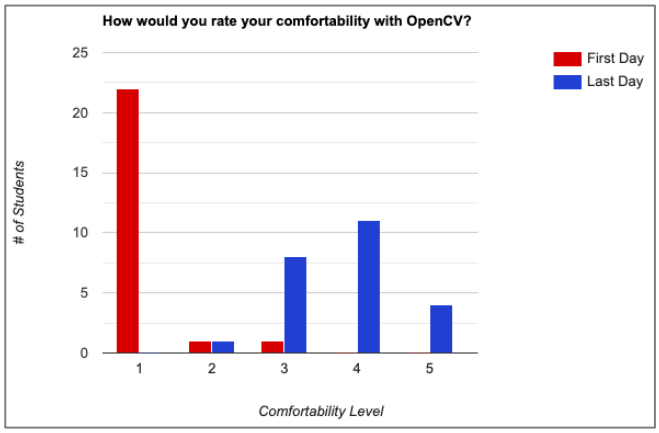

Figure 6. The student self-assessment questionnaire results for OpenCV.

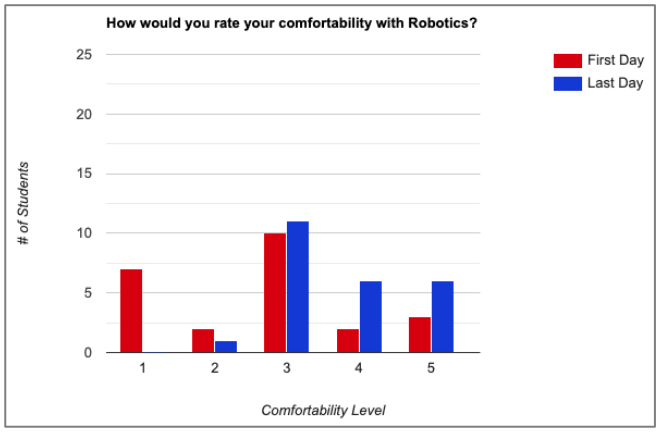

Figure 7. The student self-assessment questionnaire results for Robotics.

In Figure 7, we ask the students about their comfort level with robotics. We notice that the students report various comfort levels. We note that there were some students who have been involved in robotics programs and extracurriculars in school. However, most of these programs have focused more on the mechanics side of robotics, e.g., building hardware that can be controlled manually with a pre-

programmed joystick. These programs typically do not include any programming of intelligence that executes a sensing, computing, and actuation loop. Therefore, when leaving the program, we observe that most students feel comfortable with both robotics and software programming combined.

b) Self-assessment questionnaires for teachers: We also executed a self-assessment questionnaire for the teachers, before and after our teacher training program that was run during January 2020. Similarly to the questions asked of the students, we attempted to assess the teachers' comfort level with (i) Python, (ii) OpenCV, and (iii) Robotics. The choices ranged from 1 to 5, with 1 being "not comfortable" and 5 being "very comfortable". There were 22 teachers total in the January 2020 session, however only 15 took both the First and Last Day questionnaires. Figures 8-13 only show the questionnaire results of these 15 individuals.

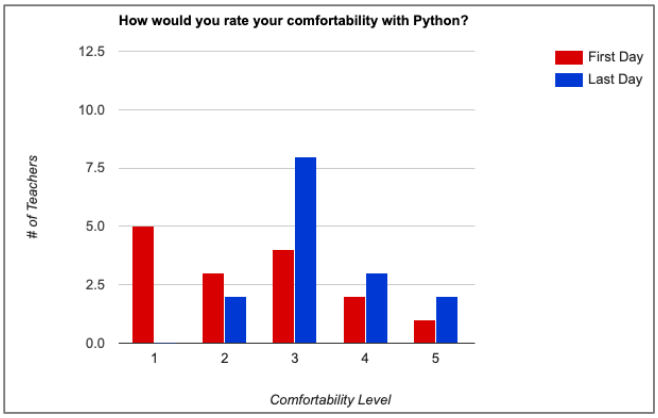

Figure 8. The teacher self-assessment questionnaire results for Python.

In Figure 8, we ask the teachers about their comfort level with Python. We notice that the teachers come with a wide range of familiarity with Python. However, most of the teachers are not comfortable with Python before starting our program. After the program we notice that most of them gain some familiarity.

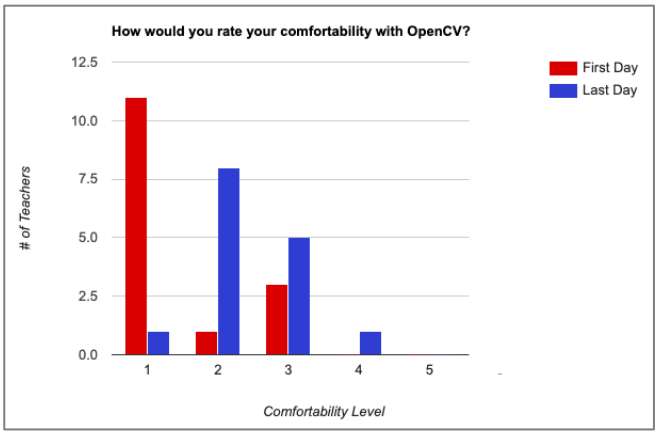

Figure 9. The teacher self-assessment questionnaire results for OpenCV.

In Figure 9, we ask the teachers about their comfort level with OpenCV. We notice that the majority of teachers ranked 1 (not comfortable) for OpenCV. While a few teachers have heard about OpenCV before, most teachers have never utilized any functionality of OpenCV. We notice that teachers become more familiar with OpenCV, even though almost all of them do not reach the comfort level above the neutral level (score of 3 out of 5) in the time frame available for the program.

In Figure 10, we ask the teachers about their comfort level with robotics. We notice that the teachers come to our program with varying comfort levels. After the program, we observe a general increase in their comfort level with robotics, after working with the RACECAR-N platform.

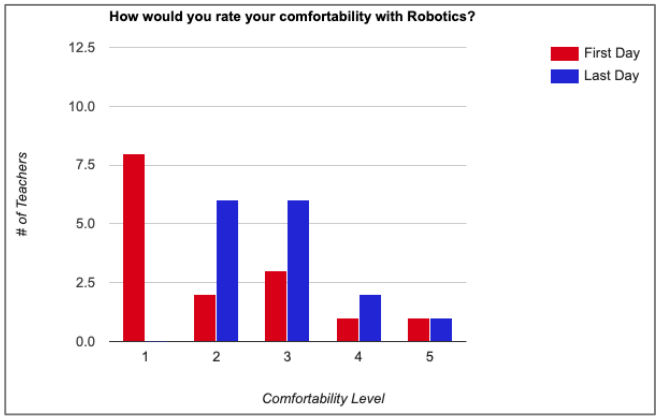

Figure 10. The teacher self-assessment questionnaire results for Robotics.

c) Teachers Feedback on Scaling: Our hope is that the course we have developed can be scaled by teachers teaching the same material in their own schools, potentially with the online materials supplied by MIT. To assess this possibility, we also asked the teachers a number of questions to get feedback on our program and its potential on middle school and high school education, which we report in Figures 11-13.

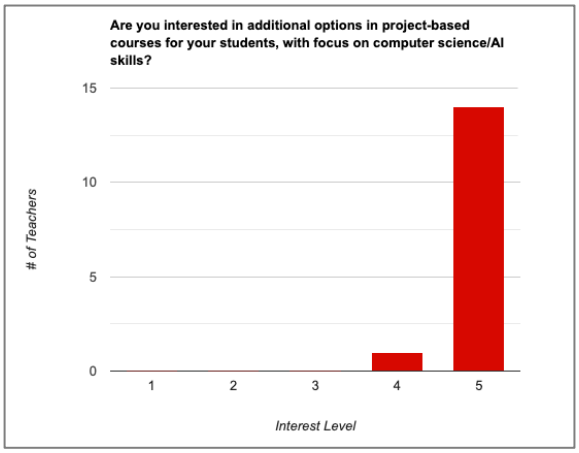

Figure 11. The teacher questionnaire result for interest in project-based courses in robotics and artificial intelligence.

In Figure 11, we ask the teachers whether they are interested in providing their students with more options in project-based courses in computer science and artificial intelligence. The choices ranged from 1 to 5, with 1 being “not interested” and 5 being “very interested”. We observe that almost all of the teachers we questioned tell us that they are very interested.

In Figure 12, we ask the teachers whether this program is suited for their school. The choices ranged from 1 to 5, with 1 being ”not suitable” and 5 being ”very suitable”. We observe that a majority of teachers report that this program is very suitable, while a few of them report that the program is not suitable.

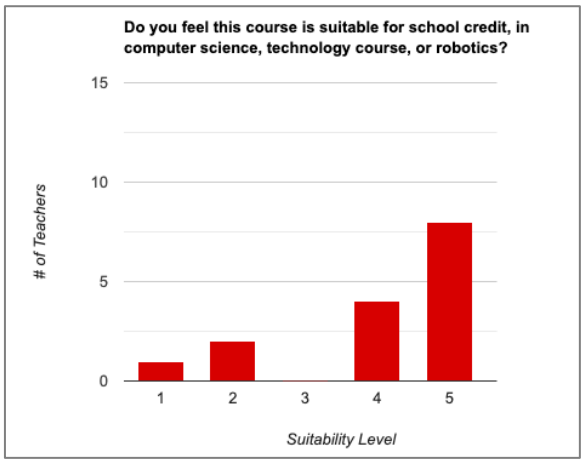

Figure 12. The teacher questionnaire result for suitability of this program in their schools.

Figure 13. The teacher questionnaire result for whether our sessions with them will allow them to teach our program to their students in their schools.

In Figure 13, we ask the teachers whether our sessions with them allow them to teach the same program in their middle schools and high schools. The choices ranged from 1 to 5, with 1 being ”not prepared” and 5 being ”very prepared”. We observe that the teachers are neutral, with a slight preference towards being able to teach the same course.

We recall that this program was a four-day program spread over four weeks on every. We believe that, with more hours invested in this training, the results might improve further for the teachers. However, providing this education to the teachers in a reasonable time frame still remains an important challenge.

Conclusions

In this paper, we presented a new middle school robotics software program. The program is powered by a mini car-like robotic vehicle designed in-house. This platform features an easy-to-use software development environment enabled by Jupyter notebooks. The students learn the basics of programming with Python, computer vision with OpenCV, and autonomous robotics. They apply their skills in a final course challenge. In the preliminary assessment of the course, we observe that the students report that they have become familiar with programming, computer vision and robotics software through this curriculum. We also ran a training session for middle school and high school teachers, who report that they are interested in teaching project-based courses that emphasize computer science and artificial intelligence, and that many of them find our program valuable. However, our training of the teachers may be relatively short, with only a few teachers prepared to teach it in their own schools at the end. Hence, training the next-generation of teachers to scale our program to middle schools and high schools across the country remains a challenge for the future.

References

[1] Karaman, Sertac et al., Project-based, Collaborative, Algorithmic Robotics for High School Students: Programming Self-driving Race Cars at MIT, Proceedings of the IEEE Integrated STEM Education Conference, 2017.

[2] C. Chalmers, “Learning with FIRST LEGO League,” *Society for Information Technology and Teacher …*, 2013.

[3] H. Fike, P. Barnhart, C. E. Brevik, E. C. Brevik, C. Burgess, J. Chen, S. Egli, B. Harris, P. J. Johanson, N. Johnson, M. Moe, and R. Olsen, “Using a robotics competition to teach about and stimulate enthusiasm for Earth science and other STEM topics,” *EGU General Assembly*, 2016.

[4] A. Saenz-Otero, J. Katz, and S. Mohan, “ZERO-Robotics: A student competition aboard the International Space Station,” *IEEE Aerospace Conference*, 2010.

[5] S. Nag, I. Heffan, A. Saenz-Otero, and M. Lydon, “SPHERES Zero Robotics software development: Lessons on crowdsourcing and collaborative competition,” presented at the IEEE Aerospace Conference, 2012, pp. 1–17.

[6] D. W. Miller, “ZERO-Robotics: a Student Competition Aboard the International Space Station,” presented at the Next-Generation Suborbital Researchers Conference, 2010.

[7] A. Saenz-Otero and J. Katz, “The Zero Robotics SPHERES Challenge 2010,” *IEEE Aerospace and Electronic Systems Magazine*, 2011.

[8] Freeman, S., Eddy, S. L., McDonough, M., et al. 2014. “Active learning increases student performance in science, engineering, and mathematics,” *Proceedings of the National Academy of Sciences*, 111(23), 8410-8415.

[9] Kolberg, E., and Orlev, N. October 2001. “Robotics learning as a tool for integrating science technology curriculum in K-12 schools.” In

31st Annual Frontiers in Education Conference. Impact on Engineering and Science Education. Conference Proceedings (Cat. No. 01CH37193) (Vol. 1, pp. T2E-12). IEEE.

[10] Welch, A., and Huffman, D. 2011. "The effect of robotics competitions on high school students' attitudes toward science." *School Science and Mathematics*, 111(8), 416-424.

[11] Herger, L. M., and Bodarky, M. March 2015. "Engaging students with open source technologies and Arduino." *In 2015 IEEE Integrated STEM Education Conference* (pp. 27-32). IEEE.

[12] Nugent, G., Barker, B., Grandgenett, N., & Welch, G. 2016. "Robotics camps, clubs, and competitions: Results from a US robotics project." *Robotics and Autonomous Systems*, 75, 686-691.

[13] Morais, I., and Bachrach, M. S. March 2019. "Analyzing the Impact of Computer Science Workshops on Middle School Teachers." *In 2019 IEEE Integrated STEM Education Conference (ISEC)* (pp. 57-61). IEEE.

[14] E. Irigoyen, E. Larzabal, and R. Priego, "Low-cost platforms used in Control Education: An educational case study," presented at the IFAC Symposium Advances in Control Education, 2013, vol. 46, no. 17, pp. 256–261.

[15] C. Chalmers, "Learning with FIRST LEGO League," *Society for Information Technology and Teacher ...*, 2013

[16] E. Danahy, E. Wang, J. Brockman, A. Carberry, B. Shapiro, and C. B, "LEGO-based Robotics in Higher Education: 15 Years of Student Creativity," *International Journal of Advanced Robotic Systems*, pp.1–16, 2014.

[17] L. E. Whitman and T. L. Witherspoon, "Using legos to interest high school students and improve k12 stem education," presented at the 33rd Annual Frontiers in Education, 2003. FIE 2003., 2003, vol. 2, pp. F3A_6–F3A_10.

[18] A. Salamon, S. Kupersmith, and D. Housten, "Inspiring Future Young Engineers Through Robotics Outreach," presented at the Proceedings of the Global Conference on Educational Robotics, 2008, pp. 1–7.

[19] E. Afari and M. S. Khine, "Robotics as an Educational Tool: Impact of Lego Mindstorms," *IJIET*, vol. 7, no. 6, pp. 437–442, 2017

[20] Tribelhorn, B., and Dodds, Z. April 2007. "Evaluating the Roomba: A low-cost, ubiquitous platform for robotics research and education." *In Proceedings 2007 IEEE International Conference on Robotics and Automation* (pp. 1393-1399). IEEE.

[21] M. J. Mataric, N. Koenig, and D. Feil-Seifer, "Materials for Enabling Hands-On Robotics and STEM Education," presented at the AAAI Spring Symposium Semantic Scientific Knowledge Integration, 2007, pp. 1–4.

[22] T. L. Crenshaw and S. Beyer, "UPBOT: A Testbed for Cyber-Physical Systems," presented at the Proceedings of the International conference on Cyber security experimentation and test, 2010.

[23] J. Kelly, J. Binney, A. Pereira, O. Khan, and G. Sukhatme, "Just Add Wheels: Leveraging Commodity Laptop Hardware for Robotics and AI Education ," presented at the Proceedings of AAAI Education

[24] Tani, J., Paull, L., Zuber, M. T., et al. November 2016. "Duckietown: an innovative way to teach autonomy." *In International Conference EduRobotics 2016* (pp. 104-121). Springer, Cham.

[25] Paull, L., Tani, J., Ahn, H., et al. May 2017. "Duckietown: an open, inexpensive and flexible platform for autonomy education and research." *In 2017 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1497-1504). IEEE.

[26] Quigley, M., Conley, K., Gerkey, B., et al. May 2009. "ROS: an open-source Robot Operating System." *In ICRA workshop on open source software* (Vol. 3, No. 3.2, p. 5).