Teaching Computer Programming to First-Year Undergraduates with a MATLAB Based Robot Simulator

By K.-Y. Daisy Fan, Cornell University

First-year undergraduates often find computer programming difficult because of the abstract nature of the processes and concepts involved. Students who are adept at learning complex historical facts, scientific theories, or mathematical principles may struggle with programming because it requires them to elucidate the process of solving a problem—develop an algorithm—and not just "crunch out" an answer.

One approach to making introductory computer programming more tangible to students is to use robotics. Robots show students their code in action, making the abstractions and algorithms concrete and the consequences immediate. For engineering majors, learning how to control real devices and work within the limitations of sensors and actuating hardware is an added benefit. Despite these advantages, robots are rarely used at the introductory level. The expense of purchasing and maintaining robots for a large class can be prohibitive, while sharing one robot among several students leads to lab-scheduling conflicts and student frustration.

In an introductory programming course at Cornell University, we use a simulator for the iRobot Create. Each of the 300 or so students in CS1112: Introduction to Computing Using MATLAB has direct access to the simulator, which was developed in MATLAB® at Cornell.

Designed to emulate the behavior and control characteristics of a real programmable robot, the simulator provides a low-cost way to bring the appeal and excitement of robotics to a large class while improving student comprehension of core concepts such as approximation and errors.

Teaching Introductory Programming with MATLAB

The primary focus of CS1112 is programming and problem solving, not robotics. We use the robot simulator to supplement and reinforce concepts traditionally taught in first-year computer science courses, including iteration, functions, arrays, randomness, and error handling.

There are several advantages to using MATLAB to teach a course such as this. First, most students enjoy working with the robotic simulator, whatever their background. Second, MATLAB makes even our non-simulator homework very visual, which helps motivate the students to complete the more basic assignments. With MATLAB, students rapidly learn to build programs that plot graphs or display other simple graphics. In another high-level language such as Java™, that would require significantly more infrastructure code to be handed out by the instructor.

Third, it is not uncommon for first-year students to become frustrated in introductory programming courses as they wrestle with algorithms, data structures, and new tools. MATLAB provides a friendlier environment than traditional programming languages because it enables students to get further into programming faster. Students see the results of their efforts earlier than they would with other languages, which reduces frustration and gives students confidence that they are capable of doing the work.

Fourth, learning MATLAB in the first year benefits the engineering students who make up more than 75% of the class and who will continue to use MATLAB throughout their studies at Cornell and beyond.

Developing the Simulator

To develop the iRobot Create simulator, I collaborated with a student, who did most of the coding, and my colleague Hadas Kress-Gazit, an assistant professor at Cornell’s Sibley School of Mechanical and Aerospace Engineering, who teaches the junior-level course on autonomous mobile robots (see sidebar).
We defined classes, applied object-oriented design patterns, and made full use of the object-oriented programming capabilities of the MATLAB language to build the simulator. The simulator incorporates a library from the MATLAB Toolbox for the iRobot Create, developed by Professor Joel Esposito and Owen Barton at the United States Naval Academy. The library translates MATLAB code into the low-level numerical commands used by iRobot Create, enabling programs written in the MATLAB language to control the robot. Because the simulator uses this library, the same MATLAB code can be used to control a real robot and a virtual robot via the simulator.

In addition to its core simulation capabilities, the simulator has four graphical interfaces. The most important one for the course is SimulatorGUI, which is used to visualize the movement of the Create robot on a map as well as the range of its sensors (Figure 1). Other interfaces include MapMakerGUI, for designing environment maps; ConfigMakerGUI, for modeling noise on the sensors and communication delays; and ReplayGUI, for analyzing and playing back autonomous navigation sequences.

![Figure 1. Path of a simulated robot, plotted on a predefined virtual map.](image)

Development of the simulator was funded by a grant from MathWorks. The MATLAB code for the simulator is available on SourceForge, and it has been downloaded more than 2500 times. The related course material is also freely available; colleagues at the University of Vermont are already using the simulator to teach a computer science course. I encourage educators at all levels to consider using it in their coursework.

**Autonomous Mobile Robots**

MAE 4180/5180 is a senior- and master’s-level robotics course in which students write MATLAB code that is first verified via the simulator and then used to control actual iRobot Create robots. By debugging their code with the simulator before coming to the lab, students can focus their lab time on the actual hardware and on the challenges of dealing with physical systems and real-world communication constraints.

In the junior-level course, students learn concepts such as localization (how the robot knows where it is in the world), mapping (how the robot finds out what its environment looks like), and motion planning (how the robot figures out where to move and how to get there). They then apply these concepts in hands-on projects with the simulator and real robots. The course concludes with a competition in which the students program the robot to navigate within a map, an exercise requiring both localization and motion planning.

The course imposes a heavy workload, but students report that the load is manageable—and more enjoyable—because MATLAB enables them to concentrate on applying new techniques as they develop their algorithms instead of on low-level programming details. The course also serves as a springboard for careers in engineering that require reasoning about tradeoffs and constraints.
Programming Exercises Using the Simulator

In CS1112, after students have learned the basics of computer programming and the MATLAB language, they complete four exercises using the robot simulator.

In the first exercise, the students manipulate the robot using manual control keys and then run a simple program that drives the robot forward, rotates it 90 degrees, and then drives it forward again. When the students are comfortable with these tasks, we ask them to extend the control program to make the robot complete a square. At this point, the concept of approximation error becomes much less theoretical, because the virtual robot does not make a precise 90-degree turn when commanded to do so. The students see that the apparently simple task of getting the robot to complete a square requires more thought—and more code—than they expected.

In the second exercise, students write a MATLAB control program to make the robot wander randomly, advancing more often than retreating, until it bumps into a wall (as detected by the robot’s bump sensor and infrared wall sensor). Here, the students apply several programming concepts, including user-defined functions, pseudo-random numbers, and indefinite iteration.

In the third exercise, the students program the robot to systematically scan the floor of a rectangular room using a sensor that measures reflectivity. When the sensor detects a dark marking on the white floor, the program registers the location using coordinates obtained from the robot’s global positioning sensor. When the robot has completed its reconnaissance, which ideally uses parallel traverses of the floor area, the students produce a MATLAB scatter plot of the gathered data (Figure 2).

![Figure 2. Left: The original floor pattern used in the third exercise. Right: A student’s scatter plot of discrete data collected by the robot.](image)

This exercise teaches the classic design tradeoff between efficiency and reliability. Students learn that the most efficient path, which would include no overlap of the parallel traverses, leads to poor results because of the limited accuracy of the sensors and the limited precision of the motors used to turn and position the robot. To ensure more reliable floor coverage they must adjust the algorithms, adding redundancy by increasing the overlap between traverses.

In the final exercise, students build upon the random walk and reconnaissance exercise and incorporate several programming concepts, including two-dimensional arrays and file input and output. The students’ programs record the robot’s travel during a random walk and save this data to a file. The students write MATLAB routines to analyze the recorded data and produce a color map of the floor, using color intensity to indicate how long the robot spent in each area of the environment (Figure 3).
Results and Next Steps

At the end of the semester, students complete a survey on their experience in the course and with the simulator. In CS1112, where the majority of the students are engineering freshmen who have not declared their majors, the simulator has increased student interest in computing: On a recent survey, more than 40% reported greater interest in computing skills from assignments that involved the simulator compared with assignments that did not. About two-thirds of all students reported that the simulator helped increase their understanding of the concepts of approximation and errors. On the final exam, one question tested the students’ understanding of this concept. There was a strong correlation between the students’ overall exam score and their use of simulator-based exercises as examples in answering this exam question. The students also reported that they liked seeing their code in action via the simulator. They requested an enhancement to the simulator that would enable the robot to move faster than a real Create robot. Accordingly, we’ve adjusted the simulator to enable faster than real-time simulations.

We continue to develop and expand the material covered in CS1112: Introduction to Computing Using MATLAB. Although the simulator was developed using object-oriented design principles, until now these principles have not been taught in the course. We will incorporate object-oriented programming with MATLAB into the next CS1112 class, and will continue to develop the simulator-based exercises and the simulator itself, adding new sensors as they become available in hardware.

About the Author

K.-Y. Daisy Fan is a Senior Lecturer in the Department of Computer Science at Cornell University. She is a co-author of Insight Through Computing: A MATLAB Introduction to Computational Science and Engineering. Her research interests include collaborative learning methodologies and technologies in engineering education, and the application of systems analysis techniques for water resources and environmental problems.
Products Used

- MATLAB

Learn More

- MATLAB Based Simulator for the iRobot Create
- CS1112 - Introduction to Computing Using MATLAB
- Classroom Resources: Programming and Computer Science

See more articles and subscribe at mathworks.com/newsletters.